FPGA MODULAR BASED IMPLEMENTATION OF A GRAYSCALE MORPHOLOGICAL CAMERA FOR REAL-TIME APPLICATIONS

DENNIS M. FIFIELD

FPGA Modular Based Implementation of a Grayscale Morphological Camera for Real-time Applications

by

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

March 2013

St. John's Newfoundland

Abstract

The purpose of this thesis is to develop an FPGA Modular Based Implementation of a Grayscale Morphological Camera for Real-time Applications. A key application is the development of an intelligent camera that satisfies the real-world constraints associated with autonomous vehicles: real-time performance, low power operation, and limitations on weight and size.

This intelligent camera consists of an image sensor and a programmable logic device called an FPGA (Field Programmable Gate Array) that serves as a reconfigurable hardware preprocessor. This preprocessing can filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. Processing functions on the preprocessor include histogram generation, edge detection and mathematical morphological operators. Although computationally expensive, mathematical morphological operators are a class of powerful image processing tools for applications such as detecting small targets in cluttered environments, as might be required by an intelligent camera.

FPGA devices have a significant advantage over serial CPUs and DSPs due to their extremely high throughput rate and ability to exploit parallelism. This research aims to make use of these advantages with the development of a modular-based approach to implement grayscale morphology. Since the process of capturing the image is independent of the image processing, this modular approach has the flexibility and scalability of design to allow for non-fixed algorithms that utilize morphological operators with weighted structuring elements of variable size and shape. This is an important first step in the development of an intelligent camera.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Nick Krouglicof for having confidence in my abilities and providing me with the excellent opportunity to work as part of his team. I am sure it would have not been possible without his help. I also like to thank Dr. Kaaren May for her extensive and valuable feedback throughout the process. Her attention to detail has without a doubt made this thesis a much better one. Finally, I would like to thank Memorial University of Newfoundland.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

From the discovery of the RMS Titanic in 1986 to the recent volcanic eruption in Iceland and the current containment operations undertaken by British Petroleum in the Gulf of Mexico, autonomous and remotely piloted vehicles have had revolutionary roles in making technological history over the last 25 years. These unmanned vehicles, that include autonomous underwater (AUV), surface (USV), aerial (UAV) and ground vehicles (AGV), are based on the same fundamental core technologies as all contemporary mechatronic systems; i.e. sensors and actuators, system dynamics, modeling, controls, analog and digital electronics, embedded systems, interface electronics and real-time programming. As the capabilities of unmanned vehicles continue to develop, there will be an ever increasing need to develop advanced mechatronic systems that provide increased vehicle autonomy and reliability. In this context, the long-term objective of this research is the development of an intelligent camera for autonomous vehicles with an emphasis on design to satisfy the real-world constraints. These constraints include achieving real-time performance given the limited process-

ing resources available on a mobile platform as well as low power operation, weight and size limitations, high reliability, and environmental considerations. The innovation lies in applying the mechatronics design approach in order to achieve a high level of system integration (including sensing, acquisition and processing functions), as well as to provide an unprecedented level of intelligence within the sensor platform (ability to process data and make high level decisions locally) and versatility (core technologies are applicable to air, water, and land vehicles).

As high computational requirements are the nature of computer vision and image processing applications, real-time performance is not a trivial accomplishment. Real-time performance refers to operating within the data rate of the camera. It is even more difficult when the nature of a specific application limits the size of the computing system, or the amount of power that can be consumed.

The proposed image processing functions implemented on the programmable logic devices are not computationally intensive algorithms, meaning that algorithms suitable for FPGA implementation do not have high data dependencies or have a huge resource requirement for addition and subtraction [5].

Preprocessing algorithms can significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. Moreover, the subsequent task of interpreting the information content in the original image may therefore be substantially simplified. These image processing functions on the preprocessor could also include low-level image segmentation and thresholding, histogram generation, edge detection and mathematical morphological operators. In order to achieve real-time performance, one of the objectives of the proposed project is the

hardware implementation of basic morphological operators in the FPGA.

Mathematical morphological operators are a class of powerful image processing tools for applications such as detecting small targets in cluttered environments, as might be required by an intelligent camera deployed on an autonomous vehicle. These operators have been selected for this research since the operations are well suited for FPGA implementation. The majority of the computation involves cycling through large amounts of pixel data and performing simple operations such as comparison, finding the minimum and maximum from a subset of data, and performing a relatively low volume of addition and subtraction operations. Mathematical morphological operators are also suitable for the fact that many operations can be combined to form other morphological operators, allowing the overall design to be modular. With a hardware modular design, the image processing system is able to take advantage of more advanced hardware techniques such as parallelism and pipelining.

1.1 Field Programmable Gate Array (FPGA)

Field Programmable Gate Arrays are devices that are typically programmed by the system manufacturer and suited for low production, low density designs. In general, FPGAs have a significantly higher production cost over other custom logic devices such as an ASIC or Application Specific Integrated Circuit [6]. The structure of an FPGA can differ greatly between manufacturers in terms of resources and capability, though the basics of how they operate remains the same. This thesis will focus on the Cyclone III device family by Altera for a basic description of how an FPGA operates and for all implementations and designs.

Using a hardware description language (HDL), FPGAs implement user-defined functionality by using a fabric of programmable connections between Logic Array Blocks (LABs) (Figure 1.1). FPGA also uses phase-locked loops (PLLs) for robust clock management, embedded multipliers, and memory blocks that can be configured as RAM, first-in first-out (FIFO) buffers, or ROM.

Each LAB consists of 16 Logic Elements (LEs), a local fabric of programmable connections, and carry chains for linking multiple LEs together. Each LE contains a four-input look-up table (LUT) that can implement any function of four variables as well as a programmable register that can be configured for D, T, JK, or SR flipflop operation [7]. This capability of linking multiple LEs and LABs together and program them into different configurations allows the user to create large and more complex combinational functions and sequential logic.

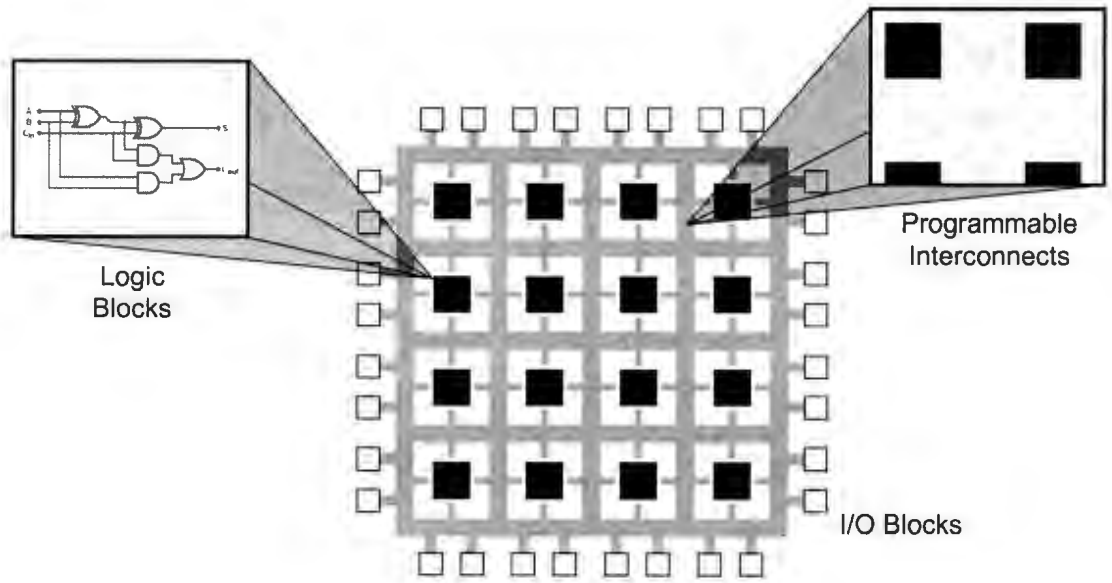

Figure 1.1: Internal Structure of an FPGA

This fabric of programmable interconnections can also interconnect to a matrix of input/output (I/O) pins. These I/O pins connect the user's design on the FPGA to any external devices present on the printed circuit board. See Alteras Cyclone III Device Handbook [7] for a more detailed description of the Cyclone III device family.

Figure 1.2 illustrates how a simplified FPGA can be configured to implement a Full Adder. Realistically, one LAB would be able to implement this simple design; though for illustration we will show the design spread over multiple LABs. Here each LAB is configured to compute the logic of each logic gate and output the result to adjoining blocks through the network of programmable connections. The values to be calculated and the corresponding result are accepted through the I/O pads on the FPGA.

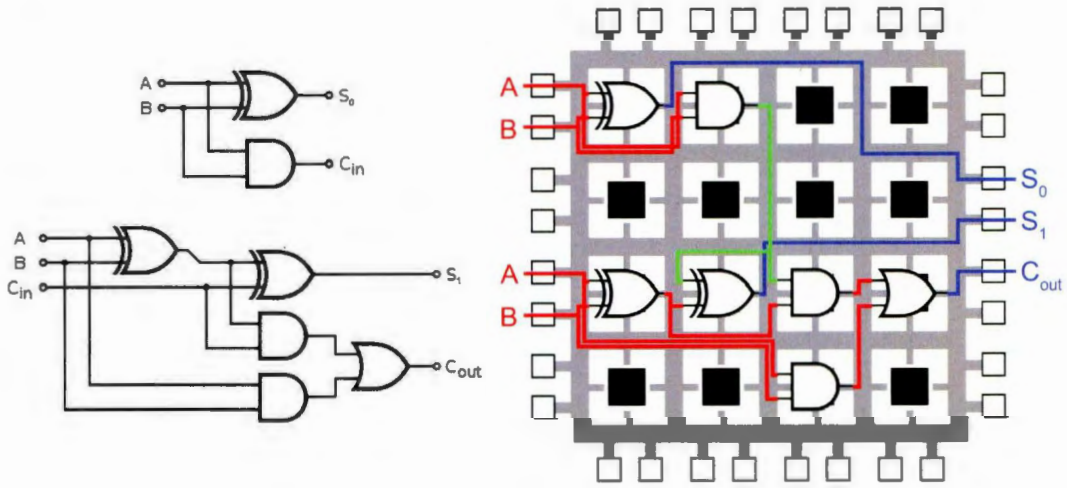

Figure 1.2: Illustration of a Simple Logic Configuration within an FPGA

1.2 Image Sensor

A wide range of high-performance line-scan and area-scan image sensors are commercially available at a level of integration that is commonly referred to as camera-on-a-chip. The two primary competing technologies are CCDs (Charged-Coupled Devices) and CMOS (Complementary-Metal-Oxide-Semiconductor) devices. Both types of sensors convert light into an electric charge, though how the signal is read from the chip is handled differently. In a CCD sensor, the light is held in each pixel as a small electrical charge and then transferred to a limited number of outputs ports. Here the charge is to be converted to a voltage, buffered, and read as an analog signal. Addition circuitry would be needed to convert the voltage into digital information. In a CMOS sensor, each pixel contains the additional circuitry to handle its own charge-to-voltage conversion. Often the sensor handles its own digitization of the voltage signal and outputs digital bits of pixel information. [8].

Traditionally CCDs have been considered superior in terms of optical performance as characterized by the quantum efficiency, dynamic range and noise. By comparison, CMOS devices offer lower power consumption and smaller system size through the integration of a number of important processing and control functions directly onto the sensor integrated circuit. These functions generally include timing logic, exposure control, analog-to-digital conversion, electronic shuttering, automatic gain control, as well as some simple image processing algorithms. In terms of optical performance, recent advances in device architecture and fabrication technology have improved the optical characteristics of CMOS devices to the point where they can effectively compete with CCDs in most applications [9].

Since the physical electronics are not in the scope of this research, it was decided to use the development board TRDB-D5M 5 megapixel CMOS camera by Terasic. The sensor is controlled by providing a clock signal within the range of 10Mhz – 70Mhz, this causes the sensor to output timing signals to facilitate pixel data capture. Faster clock frequencies will introduce greater noise on the image output. The sensor can operate in different modes depending on the application, which can be found in the image sensor technical manual [1].

The output video from this sensor is divided into frames, which are further divided into lines. By default, the sensor produces a frame that is 1,944 rows by 2,592 columns which is controlled by three output signals: *Frame Valid*, *Line Valid* and *PIXCLK*. *Frame Valid* and *Line Valid* indicate the boundaries between frames and lines, respectively, while *PIXCLK* is used as a clock to latch the output pixel data. The operation of latching the output pixels happens on the positive edge of the *PIXCLK*; one 12-bit pixel is outputted on the *DOUT* pins. The pixel is valid only when both *Frame Valid* and *Line Valid* are asserted. The timing diagram for the image sensor is shown in Figure 1.3.

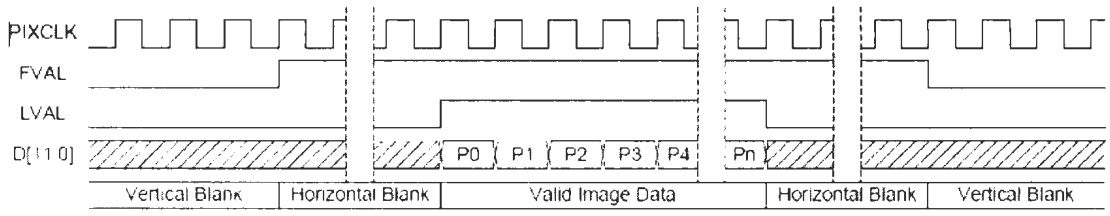

Figure 1.3: Image Sensor Pixel Output Timing Diagram [1]

The raw pixel data that comes from the sensor is a measurement of light intensities

of each pixel, which is filtered into a mosaic of specific wavelengths. This mosaic of pixels is in the form of a Bayer pattern consisting of four colors: *Green1*, *Green2*, *Red*, *Blue*. Note that even though the pixels *Green1* and *Green2* have the same color filter, they are treated as separate colors by the datapath [1]. In Figure 1.4 shows the color arrangement on the Bayer Pattern mosaic.

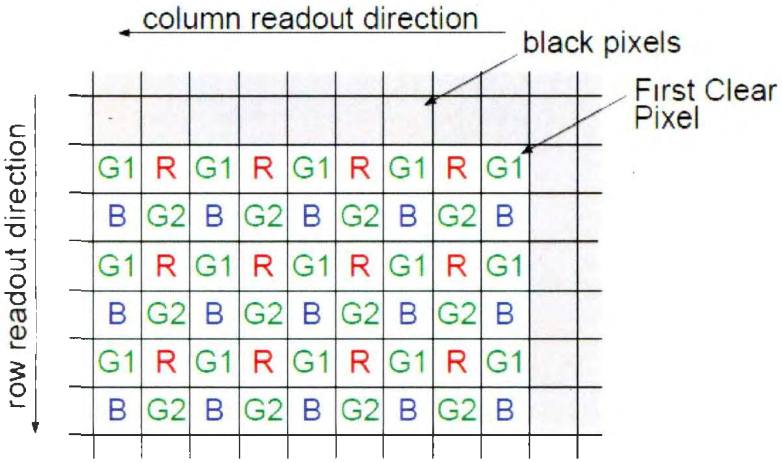

Figure 1.4: RGB Arrangement for the Raw Pixel Data from the Sensor [1]

The process of reconstructing the full color image from the incomplete color samples output from an image sensor is called *demosaicing*. The process starts by clocking the lines of pixels through a shift register, allowing the process to calculate the result of the *demosaicing* based on the current line being clocked and the previous line stored in the shift register. Nearest-neighbor interpolation is applied to determine the red, blue values of each pixel, and linear interpolation is used to generate the green value. The red, blue values are simply read from the values of the neighboring pixels and the green value is determined by taking the average of the two nearest green

pixels. The result being that RGB values for each pixel is determined by reading the value of the pixel in the same location and three neighboring pixels. This is the simplest method of interpolation to implement but is not optimal in terms of output image quality.

Chapter 2

Image Processing

2.1 Grayscale Image Morphology

Grayscale morphology operators are a set of fundamental techniques for analyzing geometrical structures within images. They are often used in industrial processes to perform narrowly defined tasks such as reading serial numbers, counting objects on a conveyor, or searching for surface defects [10]. Morphology also offers many solutions to image processing problems such as target recognition [11], finger print recognition [12], and license plate localization [13].

Although these algorithms are not mathematically intensive, they do involve iterating through large matrix structures, which can be extremely time-consuming in software implementations. However, these structures can be relatively easy to synthesize in hardware, providing real-time processing with a reduced execution time. In this case, real-time is defined as processing performed at the frame rate of the camera.

This chapter will give an introduction to image grayscale morphology, by discussing each operator in terms of its foundation in mathematical morphology, then continuing with an example to illustrate the operations being carried out in hardware, which will be covered in more detail in Chapter 4. The chapter progresses from simple to more complex morphological operators composed of other operators. For example, the operations of Opening and Closing (Section 2.1.2) are composed of Erosion and Dilation operations (Section 2.1.1).

In morphology we define two data structures: the Structuring Element S and the Input Image I. S is a matrix that defines the shape and range of a neighborhood of pixels from a defined set point within its structure (Figure 2.1). S can be any arbitrary shape and is typically smaller than the image being processed. $S_{(m,n)}$ denotes a single element on the coordinate system of the structuring element. I defines the intensity value of each pixel in the input image. $I_{(i,j)}$ denotes a single pixel on the image coordinate system.

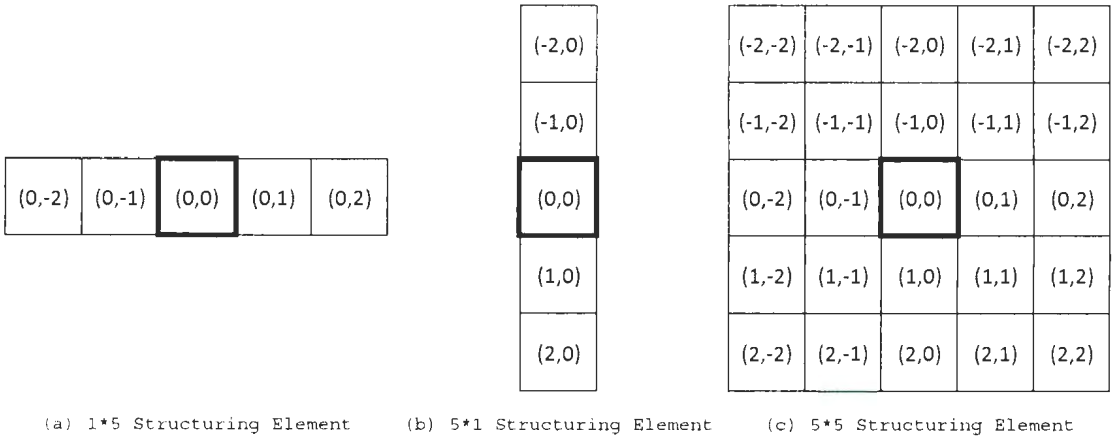

Figure 2.1: Different Shapes of Structuring Elements with Set Points

2.1.1 Erosion & Dilation

Grayscale Erosion is accomplished by taking the minimum of a set of differences in the input image expressed as:

$$I \ominus S = \min_{(m,n)\in S}\{I_{(i-m,j-n)} - S_{(m,n)} \mid (i-m),(j-n) \in I\} \tag{2.1}$$

The structuring element S is similar to a convolution mask, except that in erosion the min operation replaces the sums of the convolution and subtraction replaces the products. When the structuring element is moved from point-to-point in the input image ($I_{(i,j)}$) it results in the minimum intensity difference between the $I_{(i,j)}$ and its neighboring pixels defined by the structuring element $S_{(m,n)}$ (Figure 2.2). Note that there is nothing that restricts the coefficients (gray levels/intensity levels) of the structuring elements to be constant (i.e. "flat"). Since erosion with a flat structuring element produces the minimum intensity value of the neighboring pixels, the resulting image will be darker than the original. However, in many applications "non-flat" structuring elements are not necessary. In the simplest case, we can set the coefficients of the structuring element to 0, hence only defining the shape of the element and not its magnitude. In this case, grayscale erosion returns the minimum value of the input image within a local neighborhood defined by the structuring element.

The final output is an image that has its lower intensity pixels being spread across to its neighbors, enhancing the darker parts of the image. From Figure 2.3 you can see how different structuring elements can affect the outcome of the resulting image.

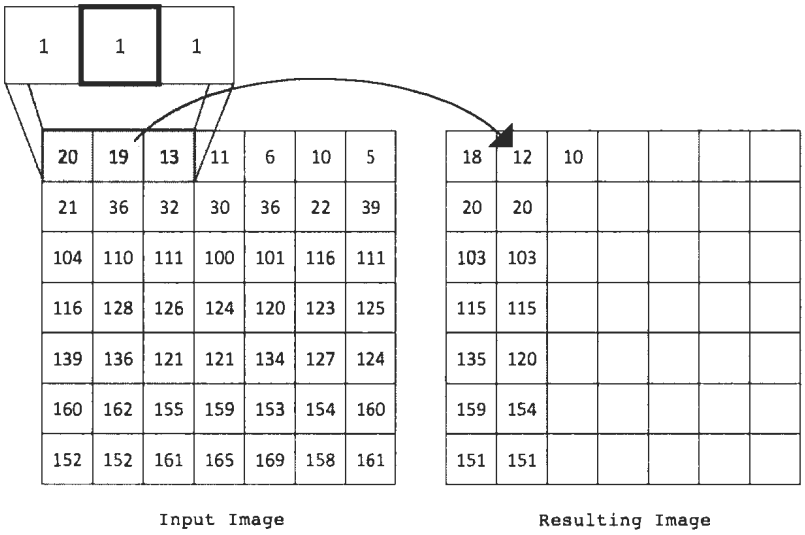

Figure 2.2: Example of Morphological Erosion on a Grayscale Image

Notice that in the case of a 1*5 structuring element the image is eroded horizontally. Likewise with a 5*1 structuring element the image is eroded vertically, but in the case of the 5*5 structuring element the image is eroded evenly throughout the image.

The equation for grayscale dilation (2.2) is similar to that of erosion, except that it takes the maximum of a set of sums rather than the minimum of a set of differences.

$$I \oplus S = \max_{(m,n) \in S} \{ I_{(i-m,j-n)} + S_{(m,n)} \mid (i-m), (j-n) \in I \} \tag{2.2}$$

Hence it has the same complexity as convolution, although instead of taking the summation of products it takes a maximum of sums [14].

Similar to erosion, a flat structuring element suffices in most applications. If all the coefficients are zero, then the result of dilation is equivalent to applying an intensity maximum operator over the local neighborhood represented by the structuring

element. The final image is similar to that of erosion, although instead of the lower intensity pixels, the higher intensity pixels are spread across to its neighbors, enhancing the lighter parts of the image. The resulting image is brighter than the original, and smaller dark details are reduced or eliminated. Likewise from Figure 2.4 you can see how different structuring elements can affect the outcome of the resulting image.

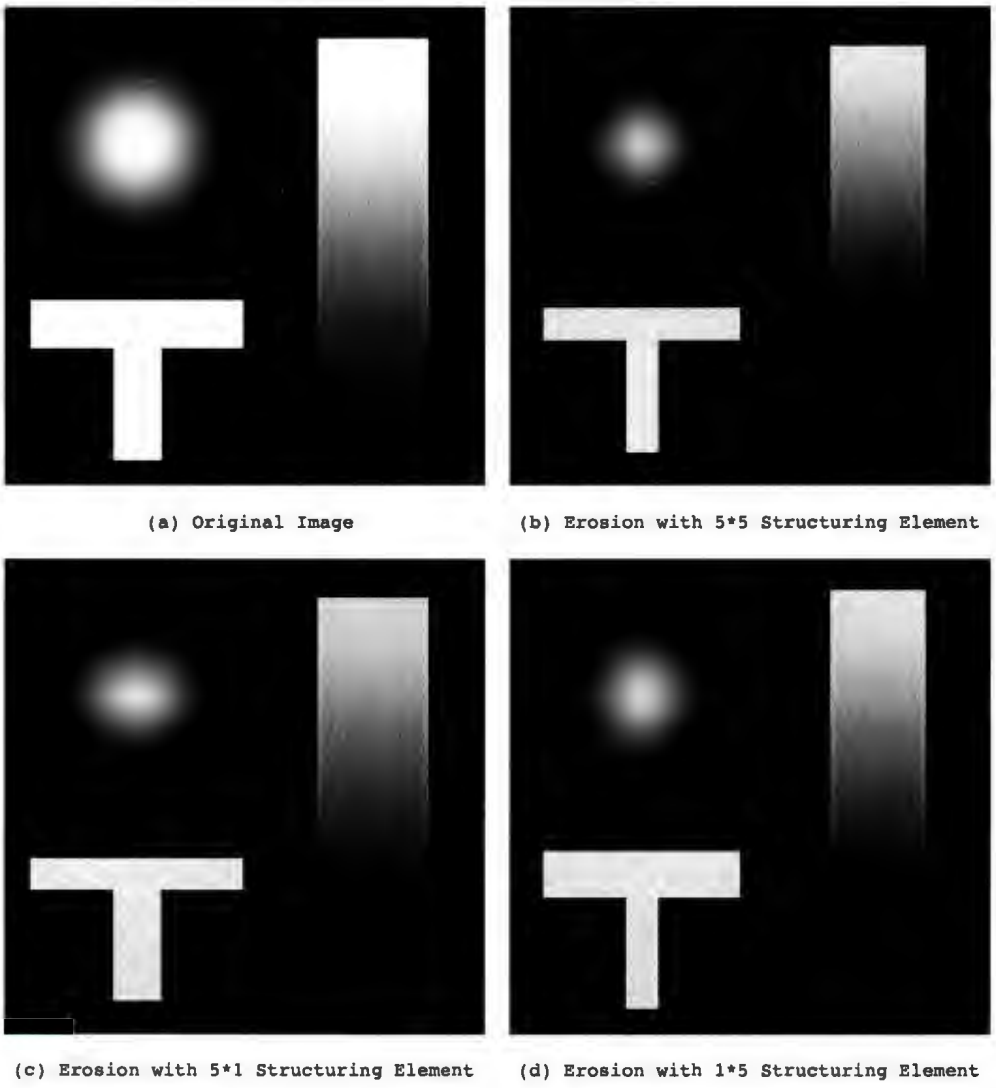

Figure 2.3: Effects of Erosion using Different Structuring Elements

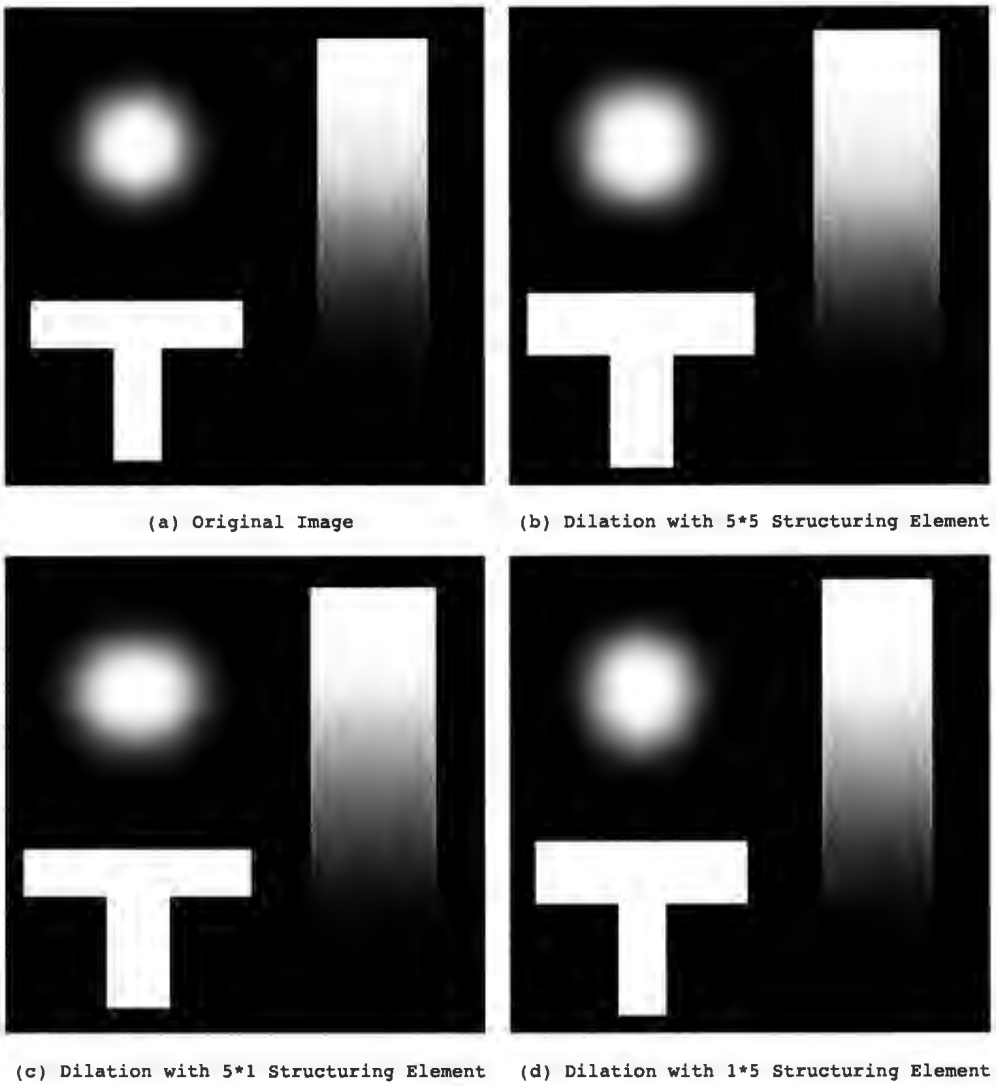

Figure 2.4: Effects of Dilation using Different Structuring Elements

2.1.2 Opening & Closing

The opening and closing morphological operations are the combination of the primary operations of dilation and erosion, though the order of the operations defines the operation being carried out. Hence, the opening of I with the structuring element S is the erosion of I by S followed by the dilation of the result by S expressed as:

$$I \circ S = (I \ominus S) \oplus S \tag{2.3}$$

The initial erosion removes the small details, but also darkens the image. The following dilation again increases the brightness without replacing the details removed by the erosion [15]. This reduces the amplitude and sharpness of all the bright peaks that are narrower than the diameter or width of the structuring element. It is usually applied to remove small light details, relative to the structuring element, while leaving the overall gray levels and larger bright features relatively undisturbed.

Similarly, the closing of I with the structuring element S is a combination of the primary grayscale morphological operations of dilation and erosion, though now the order of dilation and erosion is reversed. The closing operation of I with the structuring element S is defined as the dilation of I by S followed by the erosion of the result by S, expressed as:

$$I \bullet S = (I \oplus S) \ominus S \tag{2.4}$$

In application, the closing operation has the same effect on dark areas of the image

as the opening has on the bright areas; removing small dark details relative to the structuring element, while leaving the overall gray levels and larger bright features relatively undisturbed.

Figure 2.5: Example of Morphological Opening and Closing. (a) Original Image of size 425x448 [2]. (b) Opening of the Original Image. (c) Closing of the Original Image.

Figure 2.5 is an example of how morphological opening can be used on a grayscale image. It shows the original X-ray image of a printed circuit board (PCB) (Figure 2.5a) and the results of opening (Figure 2.5b) and the closing (Figure 2.5c). The opening of the X-ray image is performed using a flat disk structuring element of a radius of 3 pixels. Figure 2.5b shows that the bright features that are smaller, relative to the structuring element, have decreased in intensity; to the point that the smallest of the bright spots have been removed. The opening does not change the size of any feature; it only caps the higher intensities values in the grayscale image. By comparison, Figure 2.5c shows the result of a closing with a flat disk structuring element

with a radius of 5 pixels. The larger structuring element is needed with larger dark features, in order to show a result comparable to opening. It shows that the dark features that are smaller than the structuring element have decreased in intensity.

2.1.3 Top-Hat & Bottom-Hat Transform

The Top-Hat morphological transform of an image, denoted as T, and its inverse the Bottom-Hat transform, denoted as B, are defined as:

$$T = I - (I \circ S) \tag{2.5}$$

$$B = (I \bullet S) - I \tag{2.6}$$

These transforms derive their names from the structuring element that is usually used with these transforms, a cylindrical structuring element with a flat top.

As shown, the Top-Hat transform is the difference between the original image and its morphological opening, where as the Bottom-Hat transform is the difference between the original image and its morphological closing. These filters usually result in the removal of large size features, usually containing clutter, while retaining small sized features in high contrast areas, which is useful for enhancing details in the presence of shading. The difference between them is that the Top-Hat transform finds small light areas that are surrounded by relatively dark backgrounds, whereas the Bottom-Hat transform, finds dark pixel areas surrounded by relatively light backgrounds [15].

Figure 2.6 illustrates how the morphological Top-Hat transform can be used to assist in the process of *segmentation*, the process of extracting objects from the background. Figure 2.6a shows an image of individual grains of rice with non-uniform illumination, such that thresholding this image would result in a loss of some of the grains, in either the darker or lighter areas of the image. Figure 2.6b shows the opening of the original image with a flat disk structuring element with a radius of 40 pixels. This radius is large enough so that none of the objects (i.e. grains of rice) would be retained, resulting in the smoothed dark approximation of the background. Therefore by subtracting this image from the original image (i.e. performing the top-hat transform), the background becomes more uniform (Figure 2.6c). The result does still have variances in illumination in the background, but the differences between the light and dark extremes are decreased, such that thresholding would retain all the objects in the image and result in the correct *segmentation*. Figure 2.6d illustrates this, where the histogram of the original image doesn't indicate a clear separation between the background pixels and the object pixels. Whereas the histogram of the Top-Hat transform (Figure 2.6e) demonstrates a bimodal distribution, with the modes separating the background pixels from the object pixels. Thresholding at the local minimum would result in a clear segmentation of the objects. By comparison, Figure 2.7 shows an image of a running cheetah, the results of a closing operation by a flat disk structuring element with a radius of 7 pixels (Figure 2.7b), and the result of a bottom-hat transform (Figure 2.7c). It shows that the effect of the bottom-hat transform is to *segment* the black spots on the cheetah fur.

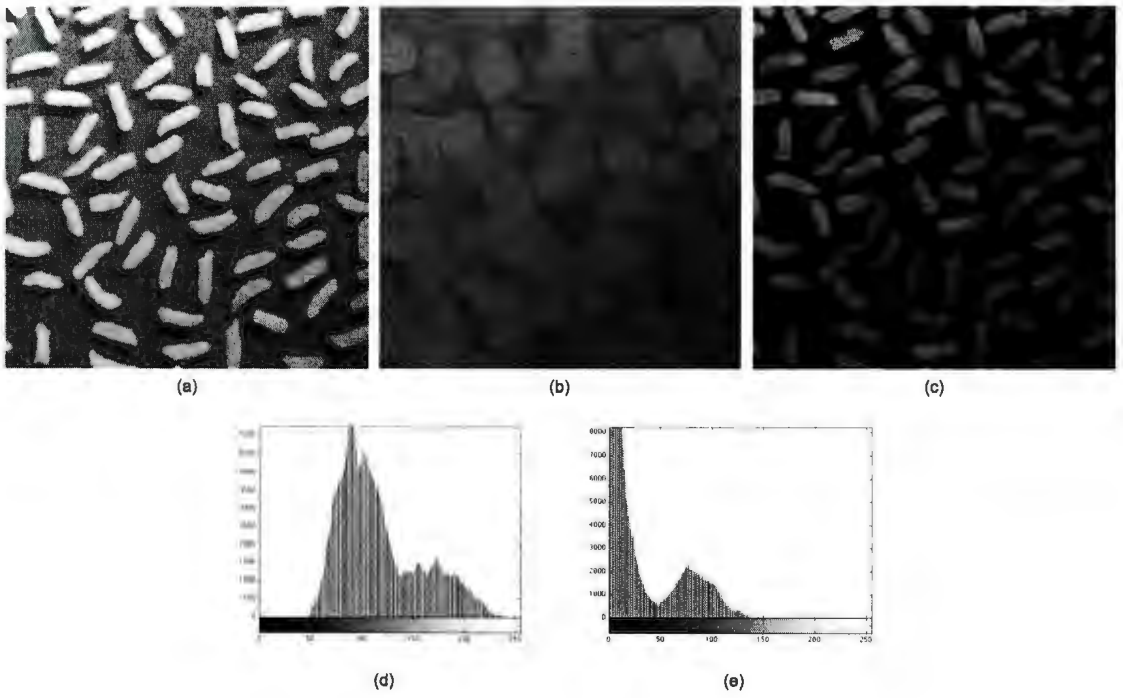

Figure 2.6: Example of Morphological Top-Hat Transform. (a) Original Image of size 600x600 [2]. (b) Opening of the Original Image. (c) Top-Hat Transform. (d) Histogram of the Original Image. (e) Histogram of the Top-Hat Transform

Figure 2.7: Example of Morphological Bottom-Hat Transform. (a) Original Image of size 600x450 [3]. (b) Closing of the Original Image. (c) Bottom-Hat Transform

2.1.4 Edge Detection

Various gradients are used in image processing to detect edges, the basic principle being that large gradients represent points where there is a rapid light-to-dark (or dark-to-light) change [16]. Edge detection or the detection of meaningful discontinuities in gray level is better defined as an area where a boundary between two regions has relatively distinct graylevel properties. The edge detection approach that is implemented here is typically referred to as the morphological gradient. This morphological edge detection or gradient is defined as the difference of the dilation of I by the structuring element S and the erosion of I by the same structuring element S, expressed as:

$$EDGE = (I \oplus S) - (I \ominus S) \tag{2.7}$$

When edge detection is used as a means of segmentation the assumption is that the regions in question are sufficiently homogenous so that the transition between two regions can be determined on the basis of grayscale discontinuities alone. When this assumption is not valid, then segmentation techniques such as thresholding and region-oriented segmentation are generally more applicable than edge detection [15].

This morphological gradient depends on the size and shape of the defined flat structuring element. Since the dilation and erosion by a flat structuring element yields maximum and minimum filters, respectively, at each point the morphological gradient results in the difference between the maximum and minimum values over the neighborhood at the point determined by the flat structuring element [15]. The

generalized structuring element for the morphological gradient is defined in Figure 2.8.

0	1	0
1	1	1
0	1	0

Figure 2.8: Structuring Element for Edge Detection

The morphological gradient can also be used in conjunction with thresholding to perform grayscale edge detection. The histogram of the gradient image is used to determine a threshold value whereby the thresholded gradient is the edge image. As with the differential gradients, the procedure is problematic owing to the non-uniformity of the gradient intensity relative to the edges [15].

Figure 2.9 illustrates a form of edge detection on an image of a medical CT scan. Figure 2.9b and Figure 2.9c shows the results of the dilation and erosion of the original with the edge detection structural element mentioned in Figure 2.8. Notice the higher intensity pixels are spread across to its neighbors in case of the dilation and the lower intensities being spread in the erosion. Taking the difference of the erosion and dilation results in the morphological gradient in Figure 2.9d.

Detecting discontinuities in image processing is important due to the fact that many discontinuities correspond to properties of depth, surface orientation, reflectance, or illumination. In the ideal case, applying an edge detector to an image may lead

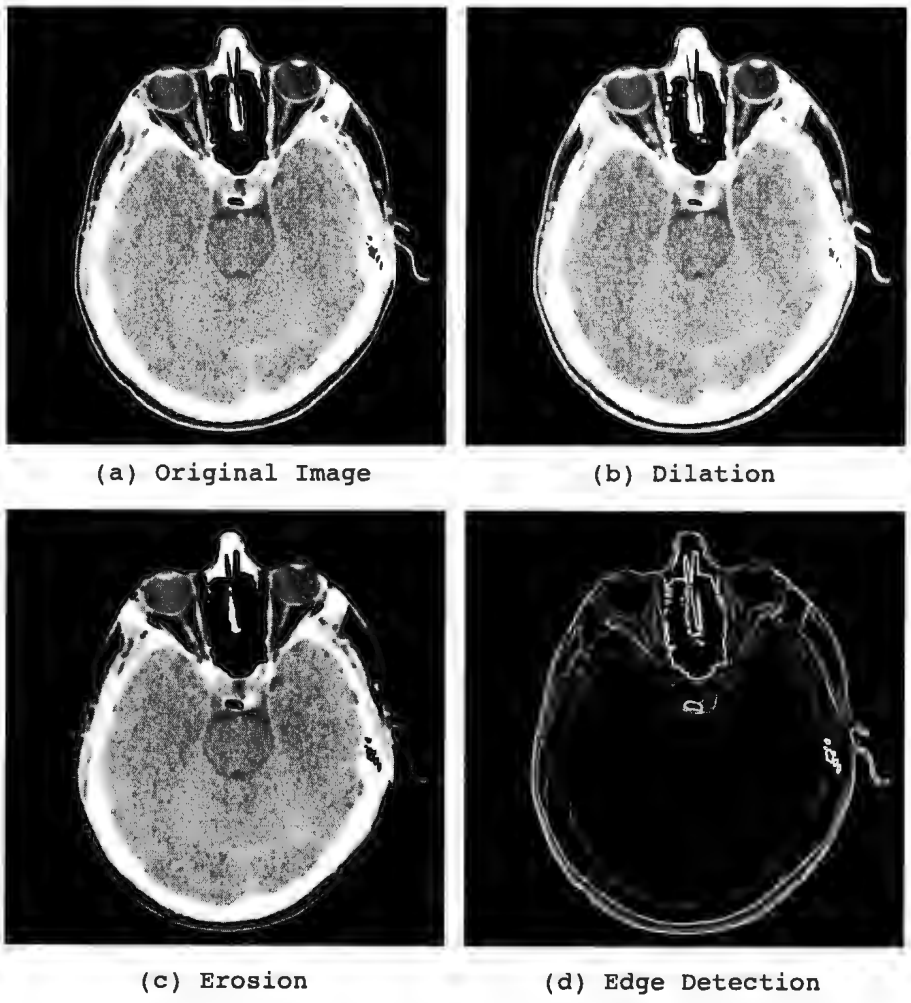

Figure 2.9: Edge Detection [2]

to a set of connected curves that indicate the boundaries of objects. Thus, this may significantly reduce the amount of data to be processed and may, therefore, filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. Moreover, the subsequent task of interpreting the information contents in the original image may, therefore, be substantially simplified.

However, it is not always possible to obtain an ideal set of edges from a real life image of moderate complexity. Edges extracted from non-trivial images are often hampered by fragmentation, missing edge segments as well as false edges not corresponding to interesting phenomena in the image, thus complicating the subsequent task of interpreting the image data [17].

2.2 Other Machine Vision Operations

2.2.1 Grayscale Image Histogram

Image histograms are a graphical representation of pixel distribution as a function of tonal variance within an image. They can be used to determine the overall exposure, indicating any detail lost due to blown-out highlights or black-out shadows.

A histogram is a simple two-dimensional graph. The horizontal axis represents the total range of tones in the image and the vertical axis indicates the number of pixels of a certain tone (Figure 2.10). The left side of the horizontal axis represents dark areas, shadows and pure black, the middle represents intermediate tones, and the right side represents light areas, bright spots and pure white. Therefore, an image of a dark or dimly lit scene with few light areas would have a histogram with most

of its pixels on the left hand side and central region of the graph. Conversely, the histogram for a bright image with few dark areas would have most of its pixels on the right hand side and central region of the graph.

More often, histograms are used to determine an appropriate threshold value, where the threshold can be determined graphically between the light and dark distributions or by using an automatic method such as Otsu's method [15]. If these distributions of dark and bright pixels are widely separated, then the image histogram is bimodal, with one mode corresponding to the dark pixels and the other to the bright pixels. Then the threshold value could be chosen in the valley of the two modes.

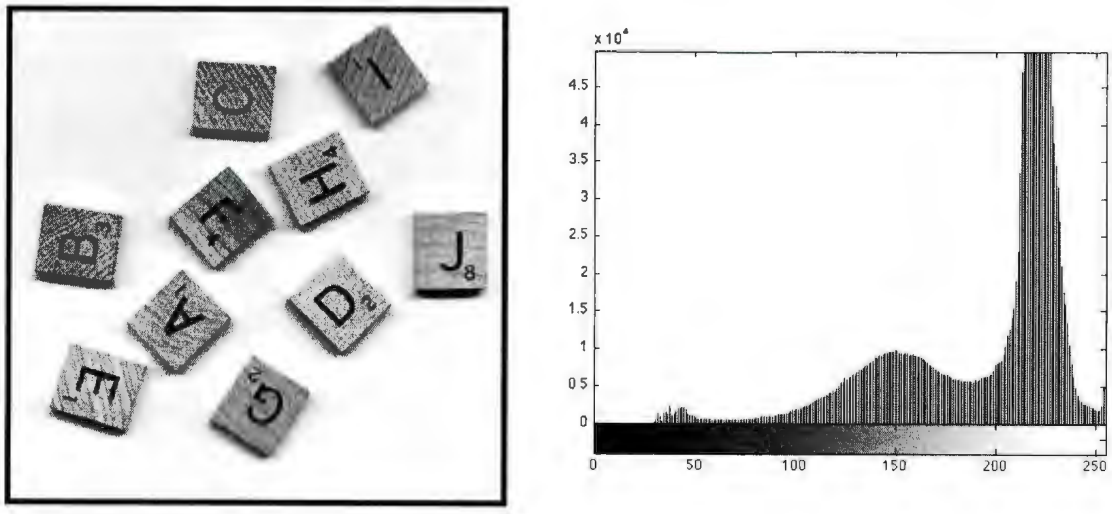

Figure 2.10: Histogram of an Image of Parts

Chapter 3

FPGA Image Processing Systems: A Review

Many researchers have investigated the use of reconfigurable hardware for digital signal processing. The extremely high throughput rate and ability to exploit parallelism give these devices a significant advantage over serial Central Processing Units (CPU) and Digital Signal Processors (DSP) for many applications. This research aims to make use of these advantages with the development of an FPGA modular-based implementation of grayscale morphological operators for real time applications. This is an important first step in the development of an intelligent camera system that could be deployed on autonomous vehicles. In particular, the use of an FPGA helps satisfy real-world constraints associated with autonomous vehicles such as real-time operation, low power consumption, and limitations on weight and size.

Several FPGA implementations of image processing algorithms have been reported in the literature [5, 18, 19, 20, 21, 22, 23, 24, 25, 26]; however, there is no indication

of a widespread acceptance of this technology within the vision community. This is probably due to the fact that most vision researchers do not have the required design expertise with FPGAs [27]. The hardware implementation of advanced image processing algorithms is also severely limited by the number of logic elements required to implement floating-point arithmetic operators. This was clearly demonstrated in a recent publication by Bannister et al. [5] that describes the implementation of a Bayesian image segmentation algorithm on a six million gate FPGA. Bannister also concluded that the complexity of the image processing algorithms should be considered; algorithms with high data dependencies or a high resource requirement for mathematical operations are not suitable for FPGA implementation [5].

The majority of implementations of morphological operations in hardware have been limited to binary morphology [19, 20, 21, 22, 23, 24, 25]. These implementations are limited mostly by the loss of information caused by thresholding the image from grayscale and are relatively simple compared to grayscale morphology; only requiring in the case of erosion a mathematical AND operation and a mathematical OR for dilation.

Other FPGA grayscale morphology implementations have been found in the literature [28, 29, 30], though many focus on different aspects of image processing and some without a clear indication of how the design has been synthesized into an FPGA. Déforges et al. [28] present an alternative approach to grayscale morphology by recursive decomposition of the structuring element. Pamula [30] presents a proposed hardware architecture for erosion and dilation as well as convolution and feature detection. Even though the design is similar to the one proposed, without a clear synthesizing of the design for FPGA, the comparison and validity of design for real

world application is difficult to determine. Many of the architectures found in the literature are limited to fixed structural elements [18] and operational frame rates that are not present in the modular based approach proposed in this research. Since the process of capturing the image is independent of the image processing, this modular approach has the flexibility and scalability of design to allow for non-fixed, non-linear image processing algorithms with weighted structuring elements of variable size and shape.

Shyang-Lih Chang et al. [18] have developed a similar approach to hardware grayscale morphology with variable structural elements and have implemented many of the same operators such as erosion/dilation, opening/closing, top-hat transform and the morphological gradient (i.e. edge detection). This system performs all morphological operators within one hardware system, an approach similar to an ALU (Arithmetic Logic Unit) in which the system is provided data and a command to select which operator to perform on the data. In comparison, the proposed modular approach in this research provides a more scalable design, such that subsequent tasks would not duplicate all other operations in hardware. The system in [18] also proposes to handle variable structural elements by scaling an internal shift register without weights, with each register multiplexed to select which elements are to be used in the operation. This allows the system to define the shape and size of the structural element, but without the ability to specify the weight of each structuring elements, thereby limiting its application.

Another FPGA real-time camera system that is similar to the proposed grayscale morphological camera system has been developed by Price et al. [24]. Price's research demonstrates the value of such a system and indicates future application of

the proposed camera architecture outside the scope of image morphology. Price et al. incorporates Altera's Cyclone FPGA with a low resolution CMOS camera to provide object detection for Unmanned Aerial Vehicles (UAV). The system detects cars and other land vehicles by performing a series of image processing operations, such as erosion, dilation, thresholding, and edge detection on a video stream without buffering the image into memory. This limits the system in the sense that it cannot exploit pipelining; hence it can only operate on one frame at a time, though parallelism is still implemented. This system detects objects independent of hue and lighting conditions, but determines points of interest partly based on the size of the object, which may limit its robustness in real-world applications.

Few systems implement image processing in FPGA hardware alone. Instead, they either use the FPGA as a datapath to a DSP [31, 32] or use a combination of both, where the FPGA is used for preprocessing and highly repetitive tasks, while the DSP is considered a low-cost solution for computationally intensive tasks [33, 26]. Another approach is given by Clienti et al. [32] that uses a System on Chip (SoC) design for image processing. SoC is a series of pipelined processors with a reconfigurable datapath similar to an FPGA. Each parallel processor computes morphological erosion, dilation, and gradient with a structuring element of 3x3. An ALU manages the results from the processors and computes basic mathematical operations. Thus, the range of morphological operators is limited and the structuring elements are fixed in size.

Many implementations rely on an external general computer to provide still images [18, 19, 21, 22, 23, 25, 31]. Ramirez-Cortes et al. [29] go as far as developing a work station for students through Matlab's graphical user interface. A student is able to perform comparisons between the results obtained from the Matlab's image

processing toolbox and the results from an FPGA-based real-time implementation. However, the ability to process still images from a computer only is a severe limitation, particularly if the object is to develop a camera that could eventually be deployed on an autonomous vehicle. A basic requirement for such a system is the ability to take image data from an active image sensor in real time (i.e. at the frame rate of the camera) instead of still images provided by a general computer. This proposed method of image collection would clearly increase the frame rate of the system and operation resources, without introducing communication delay between the system and the general computer.

In summary, other implementations of morphological operators on FPGAs are limited by a variety of factors. For example, some systems:

- are restricted to binary morphology, while few have taken on the challenges of implementing grayscale morphology.
- have not implemented a system with variable structuring elements and are restricted to flat, fixed-sized/shape grayscale operators.
- rely on general computers to provide still images to the FPGA architecture.
- rely on both an FPGA and a DSP for the image processing design.
- do not buffer the image and therefore cannot process more than one frame at a time.
- have a fixed image processing algorithm without the flexibility of a modular based approach.

- the scalability of some designs requires the duplication of additional operation not required for the image processing algorithm, thereby taking up additional resources in the FPGA. This duplication would be minimized in a modular based approach.

Chapter 4

Hardware Architecture

4.1 General System Overview

This chapter will explain the hardware architecture implemented on the FPGA. Figure 4.1 shows a basic system for simply capturing an image and displaying it through the VGA module. The following sections will explain the functionality of each module and continue adding various types of image processing modules to this basic system. For simplicity, the common clock, power, and the reset lines are not shown.

The DE0 development board from Terasic was used for the implementation of the camera system. It contains Altera's Cyclone III FPGA with 15,408 logic elements, 560 Kbits of RAM, and 4 phase-lock-loops for clock generation as well as 8-Mbyte of external SRAM, VGA connector, 40-pin header that is designed to accept a standard 40-pin IDE ribbon cable. It is also compatible with Altera's Quartus II software through the integrated USB-blaster interface circuit for on-board programming and power.

The DE0 development board was ideal for implementation since it removed design and fabrication time required for a custom printed circuit board (PCB). The development board also came with a number of daughter boards that connect to one of the 40-pin headers on the DEO which included Terasic's TRDB_D5M 5 Mega Pixel Digital Camera Package that was used for the project. This camera offers a high quality video stream at 15 frames per second at full resolution (2,592 x 1944 pixels) up to 70 frames per second at VGA standard resolution (640 x 480 pixels) with support for exposure time control for changing light conditions.

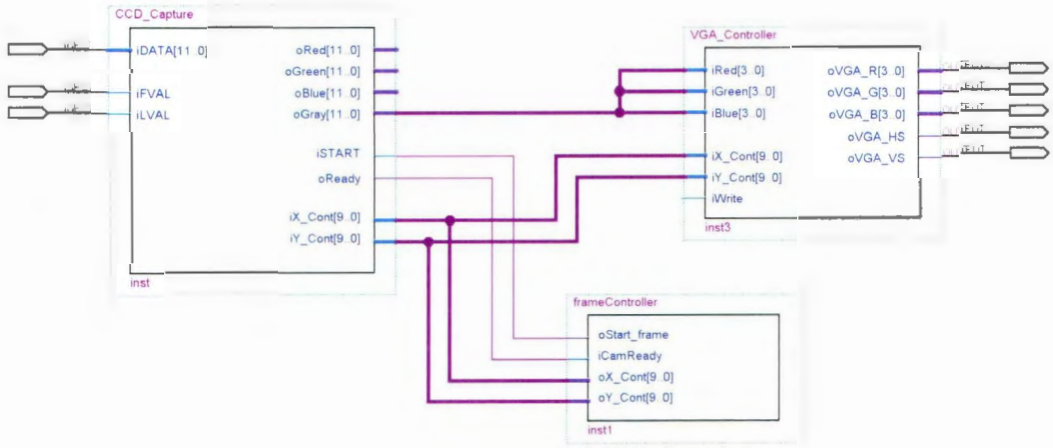

Figure 4.1: Basic System Overview

4.1.1 Camera Controller

The first stage in the process is to capture the raw pixel data from the image sensor. This is accomplished by the camera controller module that accepts the image data as well as the associated timing signals coming from the sensor. The camera module

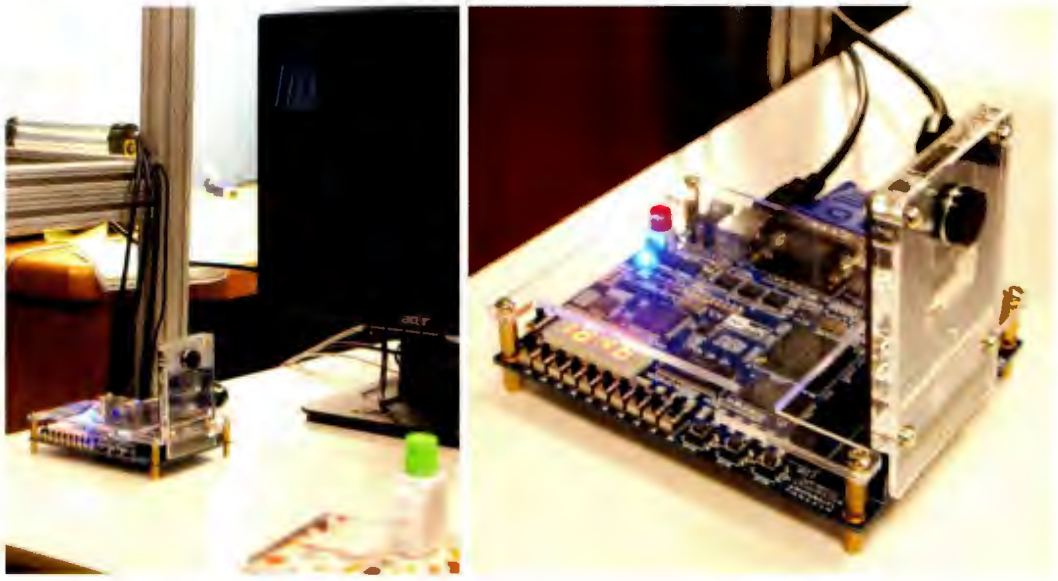

Figure 4.2: Terasic's DEO FPGA Development Board with Camera

also contains a memory buffer to store the pixel data coming from the sensor.

The image sensor is simply controlled by providing it a clock signal. The output images are divided into frames, which are further divided into lines. The *Frame Valid* and *Line Valid* signals indicate the boundaries between frames and lines, respectively. The output *PIXCLK* is used as a clock to latch the data. One 12-bit pixel is output on the *DOUT* pins on every positive edge of the *PIXCLK*. When both *Frame Valid* and *Line Valid* are asserted, the pixel is valid. While the clock signal is active, the sensor will continuously capture images and present each pixel on the positive edge of *PIXCLK*. The process of *demosaicing* starts by clocking a line of pixels and storing the values in a shift register which allows calculation of the red, green, and blue values for each pixel result based on two pixel lines at once. See Section 1.2 for a more detailed description of the image sensor's operation

Figure 4.3 shows the timing for the camera controller to accept the captured pixels from the sensor. The camera controller is initiated by a pulse on the *iStart* input signal which is latched into the register *mStart*. While *mStart* is high, the controller monitors the *Frame Valid* signal in order to synchronize the controller with the next available frame. Once the image frame is synchronized, the output signal *oReady* is driven low, indicating to the rest of the system that the camera module is currently busy. While both *Frame Valid* and *Line Valid* are asserted, the camera module will continue to latch each pixel and store it in its internal memory.

Since we are dealing with grayscale morphology, the color information is not needed. Note that the color information is still calculated and output from the camera module, but the lines are not used in this research. To convert the color information to a grayscale, certain percentages of the red, green, and blue values of the neighboring pixels are added together. The result is a representation of its luminance or luma, a value that is stored into the camera module's buffer memory. In our implementation, the conversion is based on the ITU-R BT. 601 standard, which defines luma (Y') as $Y' = 0.299\ R' + 0.587\ G' + 0.114\ B'$, where $'$ denotes gamma compression. Modern HDTV systems use the Rec. 709 standard which uses the formula $Y' = 0.2126\ R' + 0.7152\ G' + 0.0722\ B'$.

The purpose of this buffer memory is to abstract the camera module from the rest of the system. This simplifies image processing by abstracting the process of capturing and storing each image frame from the image sensor; allowing the image processing modules to focus on reading the image pixel data through the *oGray* port. This provides the pixel data at the location addressed by the pixel's X and Y coordinates within the image, *iX_Cont* and *iY_Cont*. In other words, the processes of timing,

pixel addressing, and start and ending of each frame is abstracted from the image processing modules in order to focus on the module's image processing algorithm as well as create a system design that is more flexible and robust.

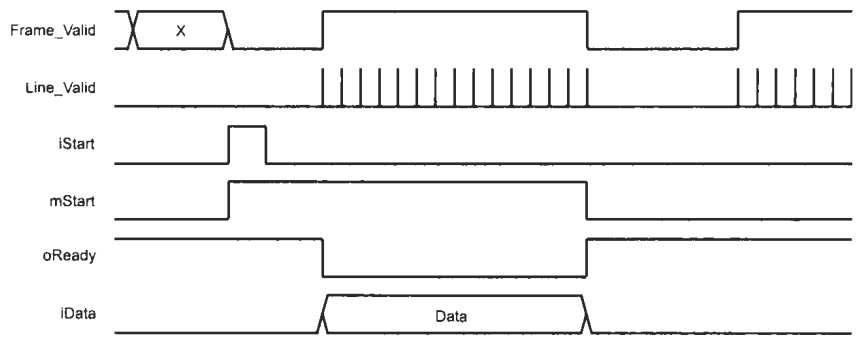

Figure 4.3: Camera Controller Timing Diagram

4.1.2 VGA Controller

The final stage in the basic camera is to output the resulting image of the morphological operations. This is done through the VGA controller module that implements the standard VGA format, with a resolution of 640 columns by 480 rows of pixels.

The VGA monitor is controlled by three analog signals that represent three color components: red, green, and blue (RGB). There are also two digital control signals: horizontal synchronization (HS) and vertical synchronization (VS). HS and VS are used to control the scan rate of each row and column displayed on the screen. HS determines the time it takes to scan a row, while the VS signal determines the time it takes to scan the entire screen. Manipulating the two synchronization signals and the RGB input produces the image.

The image is generated row by row from the top left corner of the screen to the bottom right of the screen. With each row there is precise timing, called the *horizontal blanking interval* (Figure 4.4), which has to be followed in order for the image to be displayed properly. To effectively operate the monitor at a resolution of 640x480, the synchronization control signals must run on a 25.175 MHz clock. At this frequency, each time interval is a multiple of this clock given in cycles. For higher resolutions a higher clock frequency must be used.

At the start of the process, both synchronization signals are driven high, which represents an inactive state, then HS goes low for 3.77 μs (95 cycles) called the *sync pulse*. This is the time needed for the monitor to reset the scan back to the beginning of the row. Following this, a 1.79 μs (45 cycles) high on HS, called the *back porch*, which allows time for the scan to start moving to the left again once it is at the beginning of the row. At this point, the RGB information can begin to be transmitted on the respective signal for each colour channel, one pixel at a time for 25.42 μs (640 cycles). Finally, once all the RGB pixels are clocked out, the line is driven low and the row is complete when HS remains high for an additional 0.79 μs (20 cycles), called the *front porch*. This allows time for the scan to start moving to the right to begin the next row. The entire row requires 31.77 μs (800 cycles).

The timing intervals for the VS are identical to that of the HS other than the base clock frequency which uses HS for its timing information. It starts with VS being driven low for 64 μs (2 cycles) to reset the scan to the top left corner of the screen, then the *back porch* drives VS high for 1017 μs (32 cycles). At this point, VS is kept high for 15253 μs (480 cycles), enough time for each row to be clocked out. Once this is complete, the *front porch* keeps VS high for an additional 445 μs (14 cycles).

The entire screen requires 16784 μs (528 cycles). A summary of all timing intervals is outlined in Table 4.1.

Table 4.1: HS and VS timing requirements

	Horizontal (25.175 Mhz)		Vertical (31.77 μs)	
	Time (μs)	Cycles	Time (μs)	Cycles
Sync Pulse Length	3.77	95	64	2
Back Porch	1.79	45	1,017	32
Active Pixels	25.42	640	15,253	480
Front Porch	0.79	20	445	14
Total	31.77	800	16,779	528

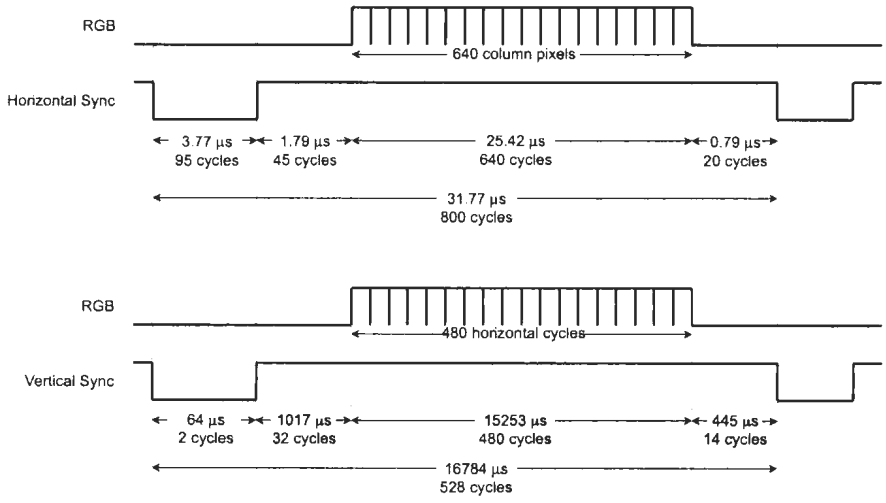

Figure 4.4: VGA Controller Horizontal and Vertical Synchronization Signal Timing Diagram [4]

The VGA module has been designed as an independent process from the rest of the system with two clock domains. This increases the efficiency of the dataflow and separates the read/write operations, which allows for the abstraction of the VGA timing process when designing the image processing hardware. A two-port RAM module that holds the captured image separates the two clock domains. Each port of the RAM accepts an independent clock: a read and a write clock. The write clock is set to the base clock of the image processing system, while the read clock is set to a 25.175 Mhz clock necessary to effectively operate the monitor at a resolution of 640x480.

Write operations are carried out by externally manipulating the input data and the control signals *iX_Cont*, *iY_Cont*, and *iWrite*. Each pixel is clocked into memory by first driving the memory's write enable high through the *iWrite* input port on the VGA module. Then the value is latched into memory by providing the pixel value on the data lines and its X and Y location within the image on the positive edge of the clock. The image is indexed as a linear array within the memory by using *iX_Cont* and *iY_Cont* as inputs.

The read operation uses two processes, which generate the HS and VS signals as well as increment two internal counters, *H_Cont* and *V_Cont*. These counters are used to address the read port on the internal RAM directly as well as keep timing for the HS and VS pulses. *H_Cont* and *V_Cont* are use to count the number of cycles that have passed for the HS and VS timing. *H_Cont* is incremented with the system clock and is reset back to zero when it has reached the total number of horizontal cycles in one row (i.e. 800 cycles). This is used for the timing interval when the HS signal must remain low to signal the horizontal *sync pulse* (95 cycles). In other words, HS

will remain low when *H_Cont* is less than 95. Once *H_Cont* reaches the sum of the *sync pulse* and the *back porch*, the pixel data can then be sent to the VGA hardware. The timing for the *front porch* interval is implicit. Operations for the VS are identical to that of HS, only that *V_Cont* is incremented only on the positive edge of the HS signal, indicating the start of each row.

4.1.3 Frame Controller

The frame controller coordinates timing for the whole system. It receives a ready signal from the camera module indicating that it has completed the previous frame and that the current frame is available. It is also responsible for initializing the start of the next frame and coordinates the pixel addressing for reading the current frame.

The frame controller facilitates the synchronization of each frame between the camera and the rest of the system. It handles two control signals: the *oReady* signal from the camera that indicates that the camera is currently busy and executes an *oStart_frame* command to start capturing the next available frame. The frame controller also handles two counters, *oX_Cont* and *oY_Cont*, used for a memory address to the pixel's data within the camera module.

This controller module is abstracted from the system to simplify image processing and image capture. With this abstraction the camera module design is simplified to focus on capturing and storing each image frame from the image sensor as well as outputting any pixel that the system wishes to read from the current frame. This controller also simplifies image processing design, allowing each image processing operation to be modular and independent for the overall system mechanics. In other

words, the processes of timing, pixel addressing, and start and ending of each frame is abstracted from the image processing modules in order to focus the module's design on the image processing algorithm and make the system's design more flexible and robust.

4.1.4 RS-485 Serial Interface

The RS-485 serial communication interface is responsible for transmitting the raw pixel data to the PC. This part of the project is not technically necessary for the intended applications of this camera, where the main goal is to analyze the video frames on board the camera and then output the appropriate control signals. However, for debugging and demonstration purposes, a picture is transferred to a general-purpose computer through an RS-485 connection, where it is displayed and stored as a common bitmap file. This is accomplished by building a RS-232 to RS-485 converter (Figure 4.5). The RS-232 and RS-485 communication protocols are the standards that have been developed by the Electronics Industry Association (EIA) to ensure reliable communication, with relatively low induced noise, differential ground potentials, transmitter/receiver impedance mismatches, and excellent biasing of the idle communication states.

The RS-232 standard permits single-ended data transmission at a relatively low data rate, 200 kbit/s at short distances of 50 feet. Despite these shortcomings (especially compared to the USB protocol) the RS-232 communication is still widely used in industry for its simplicity and ease of use. The conversion to RS-485 allows for a multi-point communication network, which in turn allows multiple drivers and

receivers to connect on a signal bus, where any node can transmit or receive data.

The start of the communication system is the signal of an RS-232 data transmission. This transmission between the master and slave device is asynchronous, which implies that there is no clock signal being transmitted with the data. The bytes are instead framed with start and stop bits being padded to the beginning and end of each byte. Communication through the RS-485 serial interface starts with the *writeEn* input set to high. Bytes are then clocked into the interface through the Data input line. Internally, the bytes are stored on the 4096 byte *First-In-First-Out* queue (FIFO). This FIFO operates with two different clock domains, meaning that the clock frequencies for read and write are different. This makes sending information through the serial communication extremely easy, since writing to the FIFO can happen at the frequency of the image processing clock, while reading happens as fast as the serial communication can handle. The user only has to ensure that the queue never overflows otherwise data may be lost. Once the FIFO detects that pixel information is stored on the queue, it sets the *rdEmpty* output to low. This causes the data on the queue to be output and sent through the serial bus automatically. This continues until the queue is emptied. The output of the FIFO is loaded into a 10-bit shift register. With the MSB bit tied to high and the LSB tied to low, these act as the start and stop bits that frame every byte that is transmitted over the serial bus.

Figure 4.6 shows the expected waveform from the RS-232 signal. While idle, the line is in a *Mark* state (+3v to + 15v). A transmission starts with a *Start* bit referred to as a *Space* (-3v to - 15v). Then each bit of data is sent down the line with the least significant bit being sent first. Then a *Stop* bit is appended to the signal to mark the end of transmission. Another *Start* bit is then sent if the transmission is

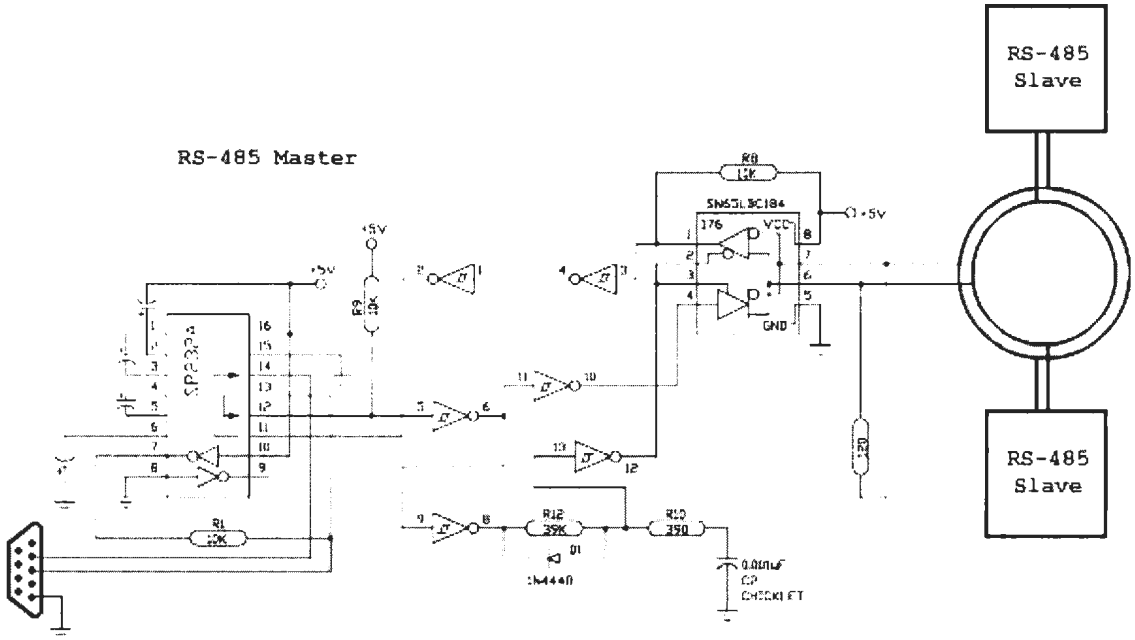

Figure 4.5: Design Circuit for the RS485 Serial Communications Transmitter / Receiver

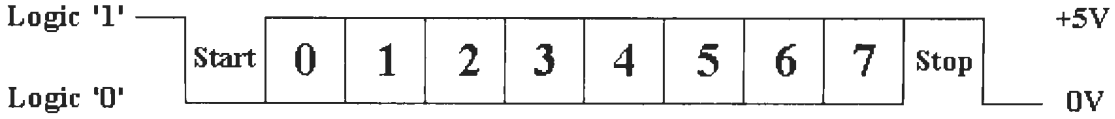

Figure 4.6: Illustration of a Framed Byte

not ended, otherwise the line is returns to idle. The receiver and transmitter are able to keep in sync with these *Start* and *Stop* bits. Should the *Stop* bit be received as a *Space*, this will cause a framing error to occur. This is common when both sides are communicating at different speeds.

The conversion to RS-485 starts with the conversion of the RS-232 signal to TTL (Transistor-transistor logic), which is used by almost all digital devices; this is done with the MAX232 converter. At this point, it is still a signal-ended transmission; to convert it to a differential signal, the signal is fed into a Differential Bus Transmitter/Receiver (SN75176).

To simplify the communication between the devices, an auto-enable circuit is used to enable the differential bus transmitter; this eliminates the need for handshaking between the master and slave. Instead the driver is in a constant state of receiving until the device is used for transmitting. The transmitting signal is also split into the auto-enable circuit causing it to double as a transmitting enable signal. Even though this will simplify the communication, care must be taken in the protocol to ensure that two devices are not transmitting at the same moment.

4.2 Image Processing

4.2.1 Erosion & Dilation

Grayscale Erosion is accomplished by taking the minimum of a set of differences between the input image and a structuring element. The structuring element defines the neighborhood of pixels around the current pixel, over which these differences are

calculated. The erosion operator is performed by finding the minimum among these set of values and placing the minimum in the corresponding output location. The erosion is complete when the structural element has been shifted over every pixel in the original image. Dilation is performed in much the same way, but instead of finding the minimum of a set of differences, dilation finds the maximum of a set of additions. See Section 2.1.1 for a more detailed description of the erosion and dilation morphological operations.

Performing erosion or dilation, therefore, requires the ability to calculate the resulting pixel based on multiple pixels that are not addressed sequentially. Figure 4.7 illustrates how this is done. It is a simple example of how a 3x3 structural element is moved from point to point in a 10x10 input image. On top of the figure there are two 8-bit shift registers in series, where the pixel data is shifted into the registers row by row, and the gray areas represent internal registers that are not readable. In order to read all the necessary pixels, a series of general-purpose registers are placed at the beginning and end of each shift register. These registers act as a parallel shift register, allowing a series of pixels to be read simultaneously. The figure shows the situation in which the structuring element is centered over the first pixel in the image, as shown in Figure 4.7a. Here, you can see that pixels labeled 1, 2, 11 and 12 can be read for the following process. As the pixel data shift through the shift register, the readable registers correspond to the structuring element being placed in every pixel in the row, as shown in the following example. All the readable pixels are added by their respected weights and then fed into a process that determines which pixels to use for the comparison based on the input *iX_Cont* and *iY_Cont*, which is the X and Y location in the input image coordinate system where the structural element mask is

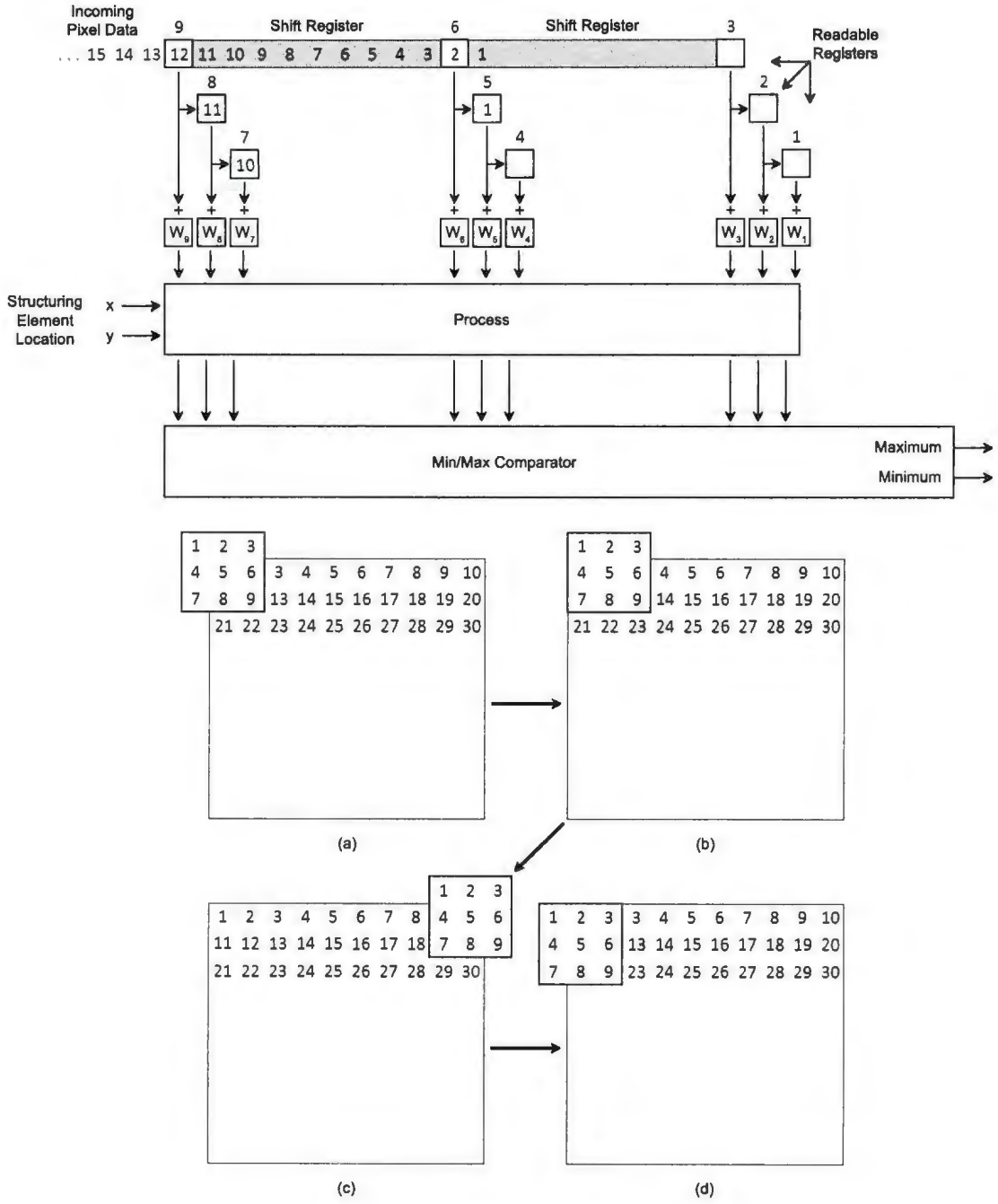

Figure 4.7: Structuring Element (SE) Dataflow (a) SE centered over the first pixel in the image (b) SE centered over the second pixel in the image. (c) SE centered over the tenth pixel in the image. (d) SE centered over the eleventh pixel in the image.

being placed. For non-square structuring elements, the process can be configured to simply ignore the inputs that don't correspond to a given structuring element shape.

The final stage is an asynchronous module compare block that compares the value of nine binary numbers, that represent the grayscale value of a set of pixels, and outputs the minimum and maximum value of the set. From the internal circuit (Figure 4.8), it is apparent that it is made up of a combination of 8-bit comparators and 8-bit multiplexors (Mux) in parallel to accept the nine binary numbers. The first stage of the comparison uses four identical sets of a single comparator and two muxes. Each comparator accepts two 8-bit binary numbers, labeled A and B, and outputs a 1-bit signal indicating if A is greater than B. A high signal indicates that this is true and a low signal indicates that this is false. This output signal is then fed into the select lines of the two muxes, determining which of the input lines will be routed to the output of the mux. On the input lines of the two muxes are the same two binary numbers, A and B, with the order of the numbers being reversed on one set of input lines on the mux. This allows the comparator to select the maximum and minimum of the two values coming out of each mux. The result is that at the output of the first stage, the nine values are grouped into two sets: a set of possible maximum values and a set of possible minimum values. All the following stages are sets of a single 8-bit comparator and a single 8-bit mux that compare the two sets of minimum and maximum values independently to find the maximum value of the set of possible maximum values and the minimum value of the set of possible minimum values.

At the end of the comparator stage, both the erosion and dilation images are created from placing the minimum and maximum outputs into the corresponding

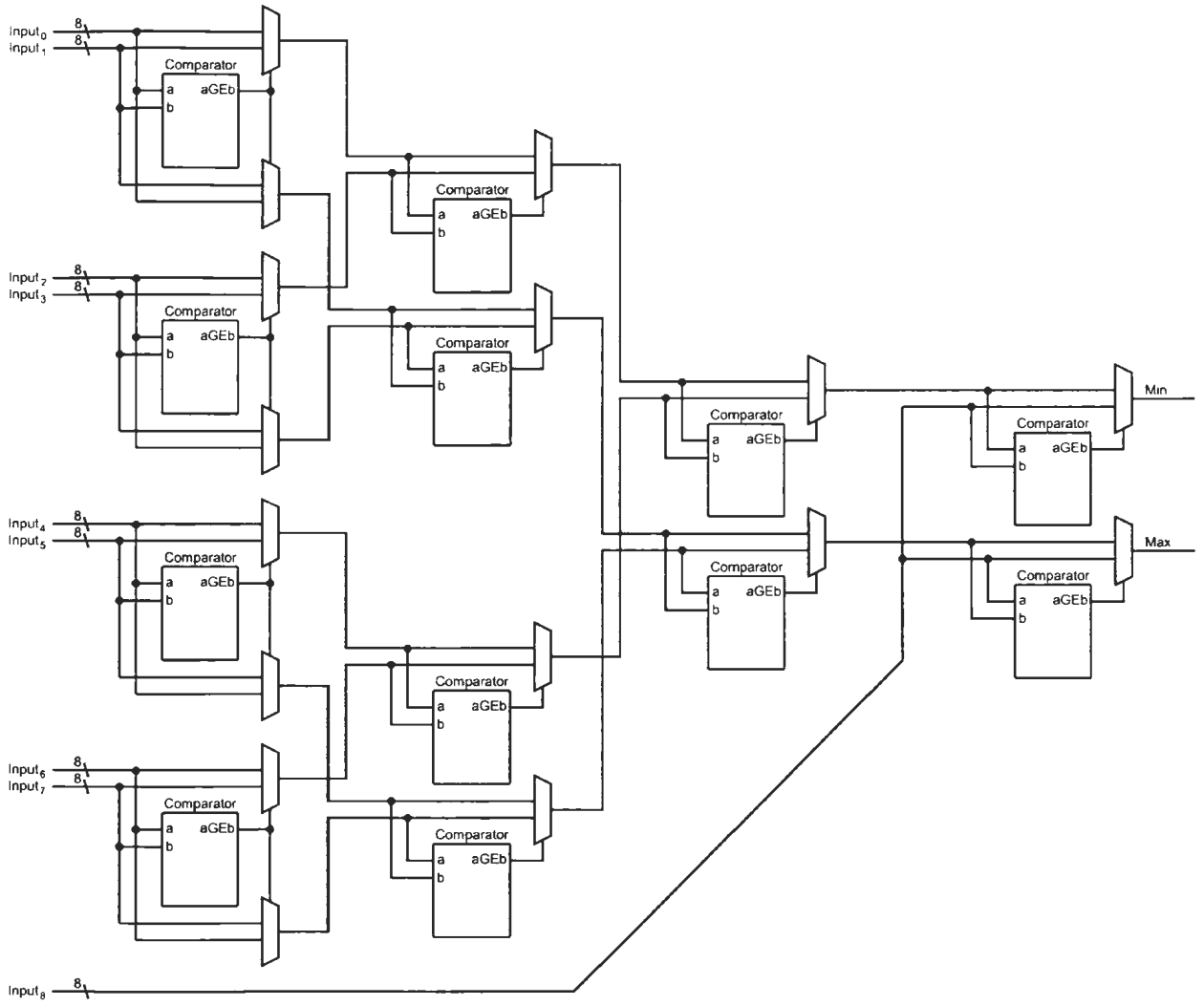

Figure 4.8: Circuit diagram for Finding the Minimum and Maximum from a Set of 8-bit Values

pixel. Increasing the size of the structuring element would simply involve adding additional shift registers and comparators in series in order to accept more pixels.

4.2.2 Opening & Closing

The opening and closing morphological operations are the combination of the primary gray-scale morphological operations of dilation and erosion, though the order of dilation and erosion defines the operation being carried out. The opening of an image is an erosion operation followed by a dilation of the result. Similarly, the closing operation has the same form with only the dilation and erosion reversed, being that the erosion is carried out on the result of the dilation. See Section 2.1.2 for a more detailed description of the erosion and dilation morphological operations.

Performing opening or closing operations requires twice as many clock cycles as other morphological operations, solely due to the operations requiring the system to iterate through the frame twice; though if you use the principles of pipelining, the throughput of each consecutive frame would be increased.

Figure 4.9 shows the opening and closing pipeline system overview. This system is derived from the basic image processing system introduced in the beginning of this chapter (Section 4.2). It is composed of a camera and VGA controller as well as two image processing modules (erosion and dilation) and frame controllers separated by a memory buffer. Note that the details of the memory buffer are not shown in this figure. The first image processing module and frame controller operate the same as the basic system. The frame controller facilitates the synchronization of each frame between the camera and the erosion module. See Section 4.1.3 for a more detailed

description of the frame controller. The erosion module accepts each pixel of the frame and performs the erosion on the frame. Before the result is outputted to the VGA controller for display, the result is stored in an intermediate memory buffer for the next consecutive dilation operation.

This second image processing module acts the same as the first; it accepts a ready signal from the memory buffer indicating that the memory is currently busy and a start command from the frame controller indicating the start of the dilation operation on the available frame. At this time, the result is read into the VGA controller memory to be displayed on the monitor.

Since the image processing modules are abstracted and have the same interface, performing a closing operation is a simple matter of reversing the order of the two image processing blocks. This modular design and the redundant repetition of the frame controller increase the system's flexibility and robustness. Furthermore, the operation increases the efficiency, allowing a consecutive frame to be captured while the pipeline completes the previous frame's operations.

4.2.3 Top-Hat & Bottom-Hat Transform

The Top-Hat transform is the difference between the original image and its morphological opening, whereas the Bottom-Hat transform uses the morphological closing operation and the difference is reversed being the difference between the morphological closing and the original image. These filters usually result in the removal of large size features, usually containing clutter, while retaining small sized features in high contrast areas, which is useful for enhancing details in the presence of shading. The

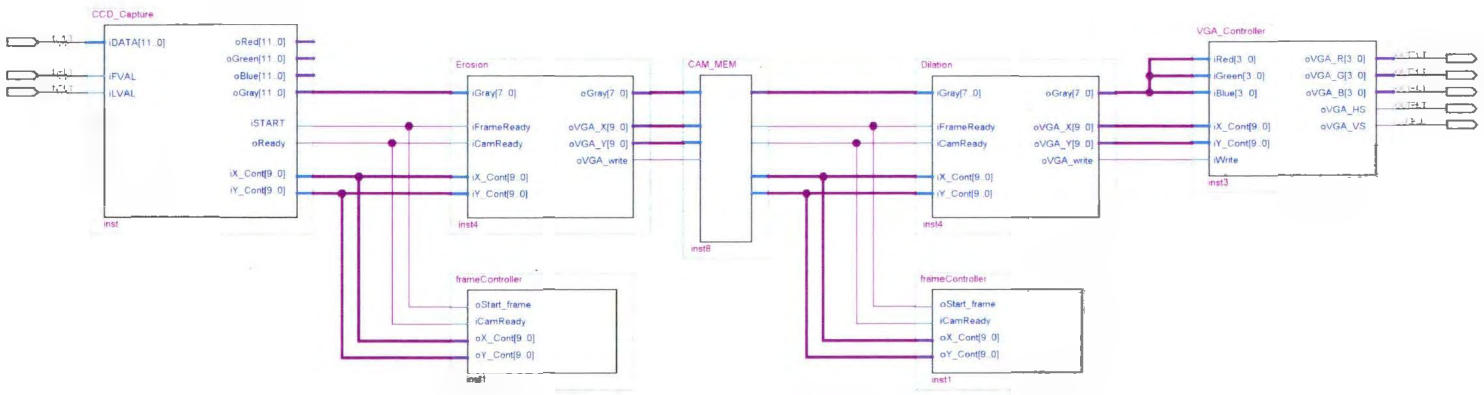

Figure 4.9: Opening & Closing System Overview

difference between them is that the Top-Hat transform finds small light areas that are surrounded by relatively dark backgrounds, whereas the Bottom-Hat transform finds dark pixel areas surrounded by relatively light backgrounds [15]. See Section 2.1.3 for a more detailed description of the Top-Hat and Bottom-Hat morphological operations.

In terms of processing hardware, this transform is a modification of the morphological operations of opening and closing, with the difference being the original image must be preserved and carried throughout the process. This is done by introducing a set of buffer memory outside the process, as shown in Figure 4.10. To preserve the pipelining aspect of the system, multiple memory buffers could be placed in series, though for illustration, only one is shown here. This memory holds the current frame until the result is produced on the output of the Open/Close image processing module. At this point, as the pixels are presented, the difference is taken between the open/close result and the corresponding pixel in the original image. This subtraction block is a standard FPGA operation with a modification to include the case in which the result falls outside the range of a 10-bit unsigned binary number.

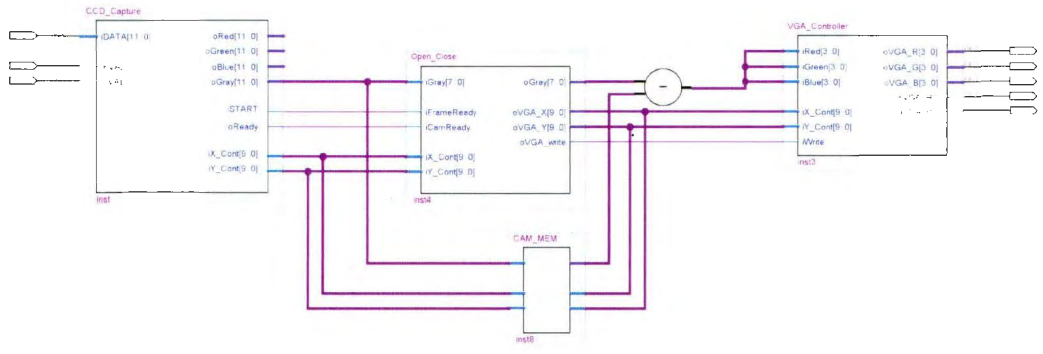

Figure 4.10: Top-Hat & Bottom-Hat System Overview

4.2.4 Edge Detection

Edge detection or the detection of meaningful discontinuities in gray level is better defined as an area where a boundary between two regions has relatively distinct gray level properties. The operation of edge detection may be one of the most important to many image processing algorithms in the sense that applying edge detection to an image may significantly reduce the amount of data to be processed and may, therefore, filter out information that may be regarded as less relevant. See Section 2.1.4 for a more detailed description of morphological edge detection.

Edge detection is implemented by placing both the erosion and dilation modules in parallel with a frame controller as presented in Figure 4.11. The frame controller manages both blocks, handling both *oReady* and *oStart_frame* signal from the camera (See section 4.1.3). Since both erosion and dilation modules have the same execution time, both results are presented to the subtraction block in the same clock interval.

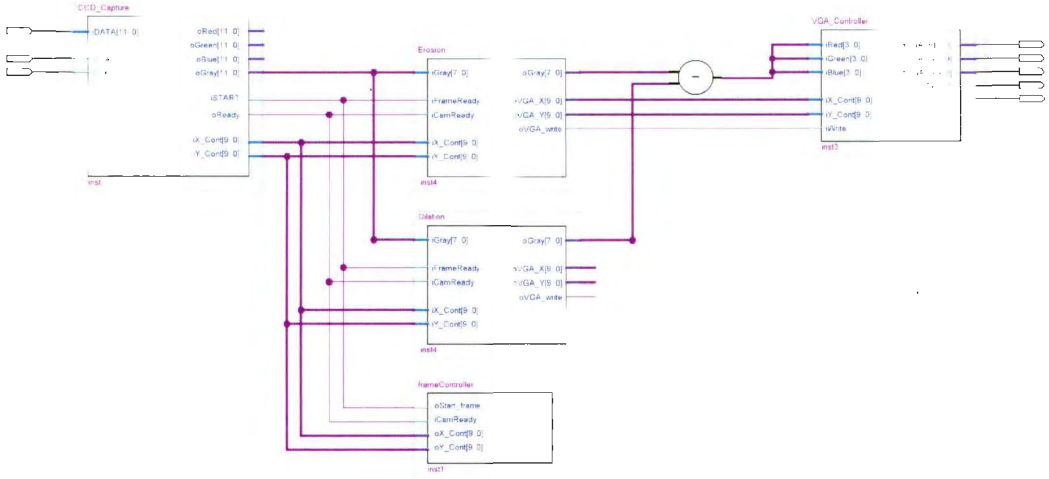

Figure 4.11: Edge Detection System Overview

4.2.5 Histogram

The histogram module is meant to build a graphical representation of pixel distribution as a function of tonal variance within the image the camera is capturing (Section 2.2.1). It consists of a programmable histogram module and three support controllers: the camera controller (Section 4.1.1), the VGA output controller (Section 4.1.2) and the frame controller. In this build, the camera controller is left unmodified, but there are some modifications to the VGA controller needed in order to output a histogram graph to the monitor, which will be discussed in this section. Figure 4.12 illustrates the entire system.

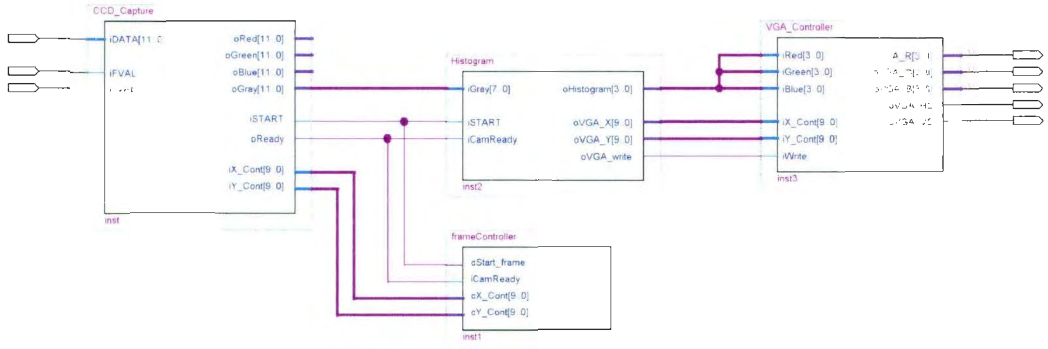

Figure 4.12: Histogram System Overview

The histogram module is implemented with a finite state machine with internal memory, which is used as an array of counters with the number of locations equal to the number of grayscale values in the image (i.e. from 0 to 255 for an 8-bit image). Therefore, as each pixel is read from the camera, the grayscale value of the pixel is used to address the memory location and increment the current value stored at that

address. The result is a count of each grayscale value in the image.

The frame controller facilitates the synchronization of each frame between the camera and the histogram module. It handles two control signals: the *oReady* signal from the camera that indicates that the camera is currently busy and executes an *oStart_frame* command to start capturing the next available frame. These two signals control five implicit states within the histogram module: *Idle*, *Wait*, *Clear*, *Build*, and *Display*. The frame controller also handles two counters, *oX_Cont* and *oY_Cont*, used for a memory address to the pixel's data within the camera module. Figure 4.13 shows the signal timing and state transitions for the system.

The flow of data through the system is dependent upon the histogram's five states. On startup, the histogram module enters the *Idle* state. This state ensures that the camera is not currently capturing the image and that both memory addresses, *oX_Cont* and *oY_Cont*, are initialized to zero. Once both conditions are true, the frame controller will signal the camera module to start processing the frame by driving *oStart_frame* to high. At this point, the system enters into the *Wait* state. The *Wait* state synchronizes the histogram module to the start of the next frame by polling the *Frame_Valid* signal until the image sensor drives the signal high to indicate the start of the frame. Once this happens, both *oStart_frame* and *oReady* are driven low in synchronization with capturing the frame. During the capture, the histogram module clears its internal memory in preparation for the frame. This will continue until the image sensor drives *Frame_Valid* low, indicating the end of the frame. At this point, *oReady* will be driven high. This will initiate the *Build* state, where the histogram module starts to build the histogram graph into memory by counting each latched data pixel on the *iGray[7..0]* bus. At the same time, the frame controller

will increment *oX_Cont* and *oY_Cont* to address each pixel in the camera module memory. This will continue until *oX_Cont* and *oY_Cont* have completely addressed all the pixels in the image, and *oStart_frame* is driven high upon completion. With the histogram complete, the histogram module will enter the *Display* state where it will generate the histogram. Once the graph is complete, the histogram will return to the *Wait* state in preparation for the next frame.

For simplicity, the histogram is displayed vertically on the screen. Taking into account that the display is generated from top to bottom and from left to right, each bar of the histogram can be drawn individually. The VGA module has been modified to display a binary image that is 200x256, where each row represents the total range of tones in the image and each column indicates the number of pixels of a certain tone. While the histogram module is in *Display* mode, the VGA's internal memory is addressed with two counters *oVGA_X* and *oVGA_Y*. As these two counters increment, the histogram's counter array is addressed with *oVGA_Y*. If the value of the counter is less than the current value of *oVGA_X*, then the module will output white pixels to lengthen the bar, otherwise black pixels as background pixels. Figure 4.14 shows the verilog code for the histogram's *Display* state.

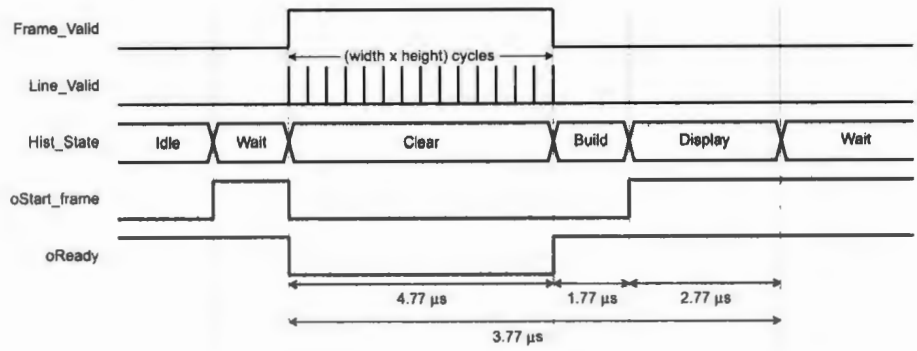

Figure 4.13: Histogram Signal Timing and State Transitions

```
if(iSTART) begin
    if(oVGA_X < coln) begin
        oVGA_X <= oVGA_X + 1'd1;
    end
    else begin
        oVGA_X <= 0;
        oVGA_Y <= oVGA_Y + 1'd1;
    end

    if(oVGA_Y > rows) begin
        oVGA_Y <= 0;
        oReady <= 1;
    end

    oVGA_write <= 1;
    if(oVGA_X <= histogramArray[oVGA_Y])
        oHistogram <= 4'h0;
    else
        oHistogram <= 4'hF;
end
...
```

Figure 4.14: Histogram display code

Chapter 5

Results

This chapter will present the results of the grayscale morphological operators implemented in this research. Due to the amount of internal memory available on the FPGA, all images in this chapter are 100 x 100 pixels in size. Since this does not affect the result of the grayscale operation, this is not considered an issue for demonstration purposes. In order for the research to have practical applications, it will be necessary to implement an external memory controller to accommodate larger frame sizes in the future.

The time latency of each frame is dependant on the type of operator being applied and is independent of the frame size. For erosion and dilation, the pixels are clocked at a rate of one pixel per clock cycle. Therefore the CMOS sensor itself is the bottleneck for the overall frame rate when one operation is being performed. This frame rate then could be dictated by configuring the CMOS sensor to the desired output that is within its capabilities. In cases where there is more than one operator, such as the opening and closing, the frame rate is halved due to the additional clock cycles

required. Note that the hardware resources needed to implement each operator are independent of the frame size. Another consideration is that the size of the structuring element is dependant on the resources available.

5.1 Erosion & Dilation

This section will demonstrate two implementations of erosion and dilation. For these basic operations any image could be used, though specific images were chosen in order to illustrate the use of different structuring elements and how they affect the result. See section 2.1.1 for a more detailed description on morphological erosion and dilation.

Figure 5.1a is a image of printed text with the corresponding erosion and dilation of this image using a 1x5 structuring element in Figure 5.1b and Figure 5.1c respectfully. Since the structuring element is asymmetrical, the operations only affect the image in the vertical direction. In the case of erosion, the lower intensity pixels are being spread across to their neighbors, enhancing the darker parts of the image; resulting in the text becoming taller and bolder. Dilation is similar to erosion, although instead of the lower intensity pixels, the higher intensity pixels are spread across to their neighbors, enhancing the lighter parts of the image; resulting in the text becoming shorter and thinner. Figure 5.2a is an image of a photograph with the corresponding erosion and dilation of this image using a 5x5 structuring element in Figure 5.2b and Figure 5.2c respectfully. This implementation is simular to the first, though in this case the effect of the erosion and dilation is in both the horizontal and vertical directions.

Table 5.1 and Table 5.2 shows the FPGA resources needed to implement both the 1x5 and 5x5 structuring elements respectfully. The two implementations only differ in the number of logic elements needed for the structuring element. The amount of internal memory used remains constant due to the fact that the image is buffered only once in both cases.

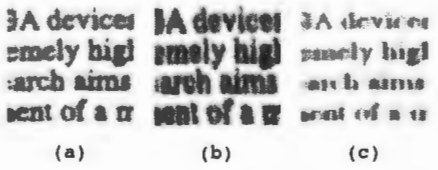

Figure 5.1: Erosion and Dilation Output from the Grayscale Morphological Camera. (a) Original Image. (b) Erosion of the Original Image using a 1x5 Structuring Element. (c) Dilation of the Original Image using a 1x5 Structuring Element.

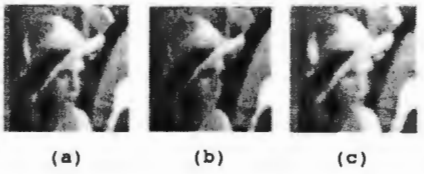

Figure 5.2: Erosion and Dilation Output from the Grayscale Morphological Camera. (a) Original Image. (b) Erosion of the Original Image using a 5x5 Structuring Element. (c) Dilation of the Original Image using a 5x5 Structuring Element.

Table 5.1: FPGA Resources required for Erosion and Dilation Operators using a 1x5 Structuring Element

	Used (Total)	Percentage
Total Logic Elements	2,902 (15,408)	19%
Total Combinational Functions	2,578 (15,408)	17%
Total Logic Registers	865 (15,408)	6%
Internal Memory Bits	336,592 (516,096)	65%

Table 5.2: FPGA Resources required for Erosion and Dilation Operators using a 5x5 Structuring Element

	Used (Total)	Percentage
Total Logic Elements	3,245 (15,408)	21%
Total Combinational Functions	2,827 (15,408)	18%
Total Logic Registers	1,129 (15,408)	7%
Internal Memory Bits	336,592 (516,096)	65%

5.2 Opening & Closing

The opening and closing morphological operations are the combination of dilation and erosion, though the order of the operations defines the operation being carried out. See section 2.1.2 for a more detailed description on morphological opening and closing.

Figure 5.3 shows an example of how morphological opening can be used on a grayscale image. It shows the original X-ray image of a printed circuit board (PCB) (Figure 5.3a) and the results of the opening (Figure 5.3b). The opening of the X-ray image is performed using a flat disk structuring element of a radius of 5 pixels. It shows that the bright features that are smaller relative to the structuring element have decreased in intensity; to where the smallest of the bright spots has been removed. The opening does not change the size of any feature; it only caps the higher intensity values in the grayscale image. By comparison, the closing (Figure 5.4b) shows the result of a closing with a flat disk structuring element with a radius of 5 pixels. The closing operation has the same effect on dark areas of the image as the opening has on the bright areas; removing small dark details relative to the structuring element, while leaving the overall gray levels and larger bright features relatively undisturbed.

Table 5.3 shows the FPGA resources required to implement opening and closing with a 5x5 structuring element. In comparison to erosion and dilation, there is a slight increase in the total number of logical elements used. This is only a slight increase due to the minimizing algorithms used by the Quartus II software.

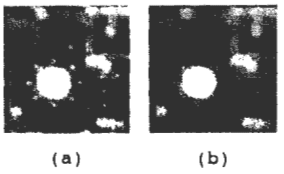

Figure 5.3: Opening Output from the Grayscale Morphological Camera. (a) Original Image. (b) Opening of the Original Image using a 5x5 Structuring Element.

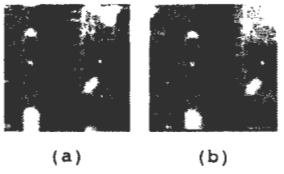

Figure 5.4: Closing Output from the Grayscale Morphological Camera. (a) Original Image. (b) Closing of the Original Image using a 5x5 Structuring Element.

Table 5.3: FPGA Resources required for Opening and Closing Operators using a 5x5 Structuring Element

	Used (Total)	Percentage
Total Logic Elements	3,795 (15,408)	25%
Total Combinational Functions	3,242 (15,408)	21%
Total Logic Registers	1,519 (15,408)	10%
Internal Memory Bits	419,760 (516,096)	81%

5.3 Top-Hat & Bottom-Hat Transform

The Top-Hat transform is the difference between the original image and its morphological opening, whereas the Bottom-Hat transform is the difference between the original image and its morphological closing. These filters usually result in the removal of large size features, usually containing clutter, while retaining small sized features in high contrast areas. This is useful for enhancing details in the presence of shading. The result of the Top-Hat transform is that it finds small light areas that are surrounded by relatively dark backgrounds, whereas the Bottom-Hat transform, finds dark pixel areas surrounded by relatively light backgrounds [15]. See section 2.1.3 for a more detailed description on Top-hat and Bottom-hat transform.

Figure 5.5 illustrates how the morphological Top-Hat transform can be used to assist in the process of *segmentation* using the morphological camera. Figure 5.5a shows an image of individual grains of rice with non-uniform illumination and the corresponding histogram generated in Matlab (Figure 5.5c). The thresholding of the original image would result in a loss of some of the grains (Figure 5.5b). By performing the Top-Hat transform on the original image, the background becomes more uniform (Figure 5.5d). The result still has variances in illumination in the background, but the differences between the light and dark extremes are decreased, such that thresholding would retain all the objects in the image and result in the correct *segmentation* (Figure 5.5e). The histogram of the Top-Hat transform (Figure 5.5f) demonstrates a clearer difference between the background and foreground pixels. By comparison, Figure 5.6a shows an image of a running cheetah with the result of a bottom-hat transform (Figure 5.6b). Note that the result clearly accentuates the black spots on

the cheetah fur from the rest of the scene.

Table 5.4 shows the FPGA resources need to implement the Top-Hat and Bottom-Hat Transforms with a 5x5 structuring element. In comparison to opening and closing, they are nearly identical; this is to be expected since the same operations are being used.

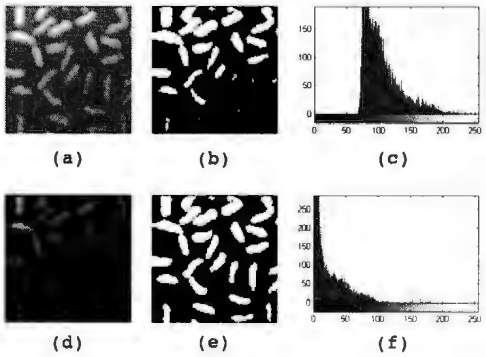

Figure 5.5: Top-Hat Transform Output from the Grayscale Morphological Camera. (a) Original Image. (b) Threshold of Original Image. (c) Histogram of Original Image. (d) Top-Hat of the Original Image using a 5x5 Structuring Element. (e) Threshold of Top-Hat Image. (f) Histogram of Top-Hat Image

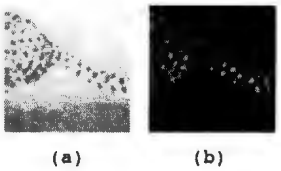

Figure 5.6: Bottom-Hat Transform Output from the Grayscale Morphological Camera. (a) Original Image. (b) Bottom-Hat of the Original Image using a 5x5 Structuring Element.

Table 5.4: FPGA Resources required for Top-Hat and Bottom-Hat Transforms using a 5x5 Structuring Element

	Used (Total)	Percentage
Total Logic Elements	3,823 (15,408)	25%
Total Combinational Functions	3,272 (15,408)	21%
Total Logic Registers	1,507 (15,408)	10%
Internal Memory Bits	434,160 (516,096)	84%

5.4 Edge Detection

Edge detection or the detection of meaningful discontinuities in gray level is better defined as an area where a boundary between two regions has relatively distinct gray level properties.

Figure 5.7a illustrates a form of edge detection on an image of a medical CT scan. Taking the difference of the erosion and dilation results in the morphological gradient in Figure 5.7b.

Table 5.5 shows the FPGA resources need to implement the morphological gradient on the camera with a 5x5 structuring element. In comparison to erosion and dilation, they are nearly identical; this is to be expected since the same operations are being used.

Figure 5.7: Edge Detection Output from the Grayscale Morphological Camera. (a) Original Image. (b) Edge Detection of the Original Image using a 3x3 Structuring Element.

Table 5.5: FPGA Resources required for Edge Detection using a 3x3 Structuring Element

	Used (Total)	Percentage
Total Logic Elements	3,155 (15,408)	20%
Total Combinational Functions	2,827 (15,408)	18%
Total Logic Registers	921 (15,408)	6%
Internal Memory Bits	334,992 (516,096)	65%

5.5 Grayscale Histogram

Image histograms are a graphical representation of pixel distribution as a function of tonal variance within an image. They can be used to determine the overall exposure, indicating any detail lost due to blown-out highlights or black-out shadows.

Figure 5.8 illustrates the implementation of real time histogram generation of an image of individual grains of rice (Figure 5.8a). Figure 5.8b and Figure 5.8c are histograms of this image using Matlab and the morphological camera respectfully. There are some differences between the two histograms due to the nature of generating an image in real time, where factors such as lighting and exposure can affect the result.

Table 5.6 shows the FPGA resources needed to implement the grayscale histogram on the camera.

Table 5.6: FPGA Resources required for Histogram Generation

	Used (Total)	Percentage
Total Logic Elements	7,375 (15,408)	48%
Total Combinational Functions	5,317 (15,408)	35%
Total Logic Registers	4,329 (15,408)	28%
Internal Memory Bits	418,224 (516,096)	65%

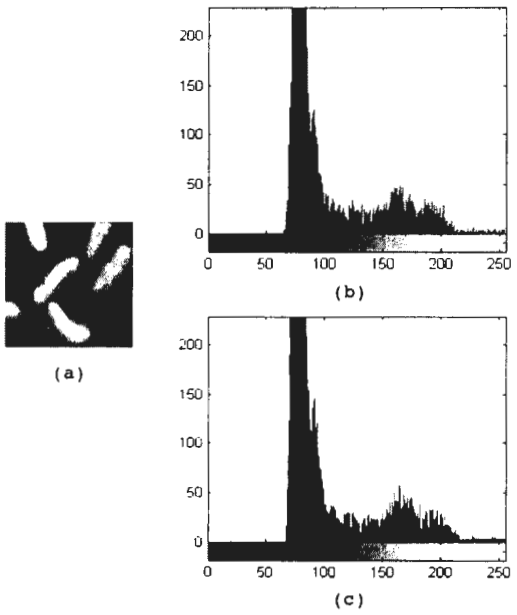

Figure 5.8: Histogram Output from the Grayscale Morphological Camera. (a) Original Image of size 100x100. (b) Matlab Histogram of the Original Image. (c) Morphological Camera Histogram of the Original Image.

Chapter 6

Conclusions

This thesis describes the modular, hardware implementation of grayscale morphological operators for demanding real-time applications. The long term objective of this research is to develop an intelligent camera that is specifically designed to satisfy the real-world constraints associated with deployment on autonomous vehicles; e.g., real-time performance, low power operation, high reliability. The image sensing, acquisition, and processing functions of the camera must all be contained within a small, lightweight enclosure. The proposed intelligent camera will consist of a CMOS image sensor and a programmable logic device (specifically a Field Programmable Gate Array or FPGA) that serves as a fast, reconfigurable hardware preprocessor. In this thesis, a number of grayscale morphological operators including erosion/dilation, opening/closing, top-hat/bottom-hat transforms and edge detection have been implemented on an FPGA. These operators all process data from the sensor in real-time. Morphological operators are commonly used for feature extraction and are therefore ideal for applications in which the role of the intelligent camera is to pick out objects

of interest based on size and shape.

Several improvements and enhancements to the hardware implementation of morphological operators have been realized in this thesis. These enhancements include:

- The hardware implementation of grayscale morphology in contrast to most real-time implementations of morphological operators that are limited to binary.
- The system is able to use (non-flat) structuring elements of arbitrary size and shape.
- The morphological camera is self-contained, not having to rely on general computers to provide still images or any other control data.
- The camera isn't reliant on DSPs or CPUs for image processing and does all operations in hardware. This gives the camera the necessary processing speed while offering the flexibility to modify the image processing algorithm with its modular design.
- The modular based design allows the system to be scalable to accommodate larger structural elements.
- Since the morphological operations are being processed faster than the image acquisition, the image processing happens in real time.
- In order to take full advantage of the FPGA hardware, the morphological operations implement buffering at each stage, which facilitates the construction of a pipeline of operations.

6.1 Future Work

Ultimately the intelligent camera system will be applied to the problem of sense and avoid for small unmanned aerial vehicles. In the envisioned two-tiered approach, the FPGA-based front-end will automatically flag a region of interest in the field-of-view of an omnidirectional camera that is used to monitor a large area. A high-resolution, narrow field-of-view camera will then be directed towards the region of interest in order to verify whether a threat exists. The FPGA camera satisfies the requirement for high reliability, low power consumption, and real-time performance. In addition to sense and avoid, potential applications of this active vision system for remotely piloted vehicles may include security and surveillance, exploration of inaccessible environments, search and rescue, event monitoring, and patrol of forest fires.

Other research projects that are adopting this approach to carry out a number of digital signal processing tasks include the development of an intelligent laser scanner which will integrate two separate modules for object detection: an intelligent camera and a 3D laser scanning system. Similar to the active vision system described previously, the intelligent camera will require a number of high level feature extraction algorithms (e.g. Fourier Descriptors, Scale Invariant Feature Transforms) to be implemented within the camera itself on an FPGA.

A list of possible future developments for the current implementation of the morphological camera include:

- Moving away from development systems and designing a custom FPGA board. This will give the flexibility of designing smaller, low power solutions, by re-

moving unnecessary hardware on current commercially available boards.

- Improvements on the visual display circuit. The current output circuit, designed by Terasic, uses a 4-bit resistor network to produce the analog video signals. This can be improved by replacing the resistor network with a digital to analog convertor (DAC) integrated circuit. This would increase the bit depth, signal resolution, and improve the signal to noise ratio. An even better solution could be to migrate from an analog display by replacing the VGA controller by a DVI or HDMI output which would be available on a higher end FPGA.

- Replacing the serial communications with a higher data rate ethernet connection.

- Implementation of other image processing techniques such as segmentation and optical flow as required for specialized applications.

- Incorporating external memory for larger frame sizes.

In addition to these possible future developments, all aspects of the system design can be further evaluated to find better and faster solutions to handle higher image resolutions and faster methods of communication with external devices. This necessitates a thorough review on the entire solution to identity potential bottlenecks.

Bibliography

[1] Terasic. *TRDB-D5M_Hardware specification*, 0.2 edition.

[2] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Pearson Education, third edition, 2008.

[3] Chris Johns. National geographic - animals, November 2011.

[4] Enoch Hwang. Build a VGA monitor controller. *Circuit Cellar*, pages 12–17, 2004.

[5] R. Bannister, D. Gregg, S. Wilson, and A. Nisbet. FPGA implementation of an image segmentation algorithm using logarithmic arithmetic. *IEEE Circuits and Systems*, 1:810–813, 2005.

[6] Dunner Kirchweg. Sparten-ii development system - introduction to FPGA technology. *trenz eletronic*, pages 1–22, November 2001.

[7] Altera. *Cyclone III Device Handbook Volume 1*, December 2011.

[8] Dalsa Dave Litwiller. Cmos vs. ccd: Maturing technologies, naturing markets, August 2005.

[9] A. Theuwissen and E. Roks. Modified CMOS processes improve image sensor performance. *SPIE Magazine of Photonics Technologies and Applications*, pages 29 32, January 2001.

[10] S.H Indera Putera and Z.Ibrahim. Printed circuit board defect detection using mathematical morphology and matlab image processing tools. In *International Conference on Education Technology and Computer*, 2010.

[11] David Casasent and Anqi Ye. Detection filters and algorithm fusion for ATR. In *IEEE Transactions on Image Processing*, volume 6, Jan. 1997.

[12] Dung Duc Nguyen, Thien Cong Pham, Xuan Dai Pham, Seung Hun Jin, and Jae Wook Jeon. Finger extraction from scene with grayscale morphology and BLOB analysis. In *International Conference on Robotics and Biomimetics*, Feb. 2008.

[13] Gisu Heo, Minwoo Kim, Insook Jung, Duk-Ryong Lee, and Il-Seok Oh. Extraction of car license plate regions using line grouping and edge density methods. In *International Symposium on Information Technology Convergence*, 2007.

[14] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1. Addison-Wesley Publishing Company, 1992.

[15] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992.

[16] Edward R. Dougherty. *An introduction to Morphological Image Processing.* SPIE Optical Engineering Press, 1992.

[17] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. Technical report, Computational Vision and Active Perception Laboratory (CVAP), S-100 44 Stockholm, Sweden, August 1998.

[18] Shyang-Lih Chang, Chih-Hao Tao, Chung-Ju Yeh, and Sei wang Chen. A hardware architecture of programmable morphological operation. In *International Conference on High-Speed Circuit Design*, Oct. 2009.

[19] Jorg Velten and Anton Kummert. FPGA-based implementation of variable sized structuring elements for 2d binary morphological operations. In *IEEE workshop on Eletronic Design, Test and Applications*, 2002.

[20] Hugo Hedberg, Fredrik Kristensen, and Viktor Owall. Low-complexity binary morphology architectures with flat regtangular structuring elements. In *IEEE Transactions on Circuit and Systems*, volume 55, Sept. 2008.

[21] Hugo Hedberg, Petr Dokladal, and Viktor Owall. Binary morphology with spatially variant structuring elements: Algorithm and architecture. In *IEEE Transactions on Image Processing*, volume 3, March 2009.

[22] Andrew J. Tickle, Fan Wu, Paul K. Harvey, and Jeremy S. Smith. Possibility of object recognition using altera's model based design approach. In *Sensors and their Applications XV*, 178. IOP Publishing, 2009.

[23] Andrew J. Tickle, Jeremy S. Smith, and Q H Wu. Development of morphological operators for field programmable gate arrays. In *Sensors and their Applications XIV*, 76. IOP Publishing, 2007.

[24] Andrew Price, Jacob Pyke, David Ashiri, and Terry Cornall. Real time object detection for and unmanned aerial vehicle using FPGA based vision system. In *IEEE International Conference on Robotics and Automation*, Orlando, Flordia, May 2006.

[25] Jaun Manuel Ramirez, Emmanuel Morales Flores, Jorge Martinez-Carballido, Rogerio Enriquez, Vicente Alarcon-Aquino, and David Baez-Lopez. An FPGA-based architecture for linear and morphological image filtering. In *IEEE*, 2010.

[26] X. Zhai, F. Bensaali, and S. Ramalingam. Real-time license plate localisation on FPGA. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 14 – 19, Colorado Springs, CO, June 2011.

[27] W.J. MacLean. An evaluation of the suitability of FPGAs for embedded vision systems. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[28] Olivier Déforges, Nicolas Mormand, and Marie Babel. Fast recursive grayscale morphology operators: from the algorithm to the pipline architecture. In *Real-Time Image Proc*, July 2010.

[29] Juan Manuel, Ramirez-Cortes, Pilar Gomez-Gil, Vicente Alarcon-Aquino, Jorge Martinez-Carballido, and Emmanuel Morales-Flores. FPGA-based educational platform for real-time image processing experiments. In *Wiley Periodicals*. Wiley Periodicals, December 2010.

[30] Wieslaw Pamula. Feature extraction using reconfigurable hardware. In *ICCVG*, volume 2, pages 158–165, 2010.

[31] Duan Jinghong, Deng Yaling, and Liang Kun. Development of image processing system based on DSP and FPGA. In *International Conference on Electronic Measurement and Instruments*, 8. ICEMI, 2007.

[32] Christophe Clienti, Serge Beucher, and Michel Bilodeau. A system on chip dedicated to pipeline neighborhood processing for mathematical morphology. In *European Signal Processing Conference*, 16, Lausanne, Switzerland, August 2008.

[33] Khursheed Khursheed, Muhammad Imran, Abdul Wahid Malik, Mattias O'Nils, Najeem Lawal, and Thornberg Benny. Exploration of task partitioning between hardware software and locality for a wireless camera based vision sensor node. In *International Symposium on Parallel Computing in Electrical Engineering*, 6, 2011.